HIGHER ORDER POISSON REGULARIZATION FOR GRAPH-BASED SEMI-SUPERVISED LEARNING

DINGJUN BIAN

Submitted under the supervision of Professor Jeff Calder to the University Honors Program at the University of Minnesota-Twin Cities in partial fulfillment of the requirements for the degree of Bachelor of Arts, *summa cum laude* in Mathematics.

March 3, 2021

Acknowledgements

I want to express my deepest appreciation to Professor Jeff Calder for advising me on this thesis and introducing me to the beauty of applied mathematics. His patient guidance and support spurred my interest in partial differential equations and their applications to machine learning. I want to thank Professor Dmitriy Bilyk for providing insights into the relation between PDEs and the calculus of variations. Moreover, I would like to express my sincere appreciation to Professor Gilad Lerman, Professor William Leeb and Professor Li Wang for reviewing this thesis and providing me with valuable feedback.

I must also thank Chieh-Hsin (Jesse) Lai for answering many of my questions on mathematics throughout the years of my undergraduate studies. I would not have gotten into the study of mathematics without his mentoring and words of encouragement. Thanks also to my study group members, including Jorge Vinicio Peña-Vélez, Zubin Lal, and Yulin An, for the countless hours studying mathematics together and exploring the beauty of mathematics.

I also have to thank Chia Thao for spending the best years of her life with me. Last but not least, I am extremely grateful to my parents for their unparalleled support throughout the years. I could not have been who I am today without them.

Abstract

Semi-supervised machine learning method is useful when labeled data is either insufficient or costly to obtain and unlabeled data is abundant. Applications of semi-supervised machine learning method includes ranges from facial and speech recognition, text classifications to medical image analysis. In this paper, we study the theoretical properties and performance of higher order Poisson regularization for graph-based semi-supervised machine learning. In particular, we prove the existence of solution for the higher order Poisson regularization problems. Moreover, we show that the solution is unique up to a constant function using a variational approach. We then develop an algorithm to solve for the higher order Poisson learning problems and compare the performance of the algorithm with other techniques such as Poisson learning and Laplace learning on common data sets for semi-supervised machine learning under very low label rates. We end the paper with a discussion on potential future working directions for higher order Poisson learning including potential ways to deal with the approximation errors when solving the higher order Poisson problems.

Contents

Introduction

Semi-supervised learning is a type of machine learning method that is between supervised and unsupervised machine learning. Unlike in supervised machine learning, where we have an abundance of labeled data and we use only the labeled data to train the model, nor like in unsupervised machine learning, where we do not make use of labeled data at all in the training step of the model, semi-supervised machine learning algorithms make use of both labeled and unlabeled data in learning tasks, such as image classification, facial recognition and medical image analysis. A common way to make use of the unlabeled data is to build graphs in order to encode the similarities between data points using edges and develop a graph-based learning algorithm to propagate the information from labeled nodes to the unlabeled ones.

One of the most widely used graph-based semi-supervised learning algorithms is Laplacian learning [17], which finds the harmonic extension of the labels on the graph through harmonic energy minimization. The performance of the algorithm is good when given a moderate (a good amount?) amount of labeled data. However, the algorithm fails to propagate the labeled information efficiently when the label rate is extremely low due to the degeneracy in the graph Laplacian with Dirichlet boundary condition. The question of how many labels are needed in Laplace learning was studied recently in [6]. Many methods have been proposed to address low label rate problems in semi-supervised learning over the past decade including higher order Laplacians [16], p-Laplacian methods [1, 2, 8, 9, 11, 15], reweighted Laplacians [5, 14], the centered-kernel method [12, 13], volume constrained MBO [10], and most recently Poisson learning [4].

The results in [4] show that Poisson learning performs better than all other methods except for the higher order methods which were not considered. In this paper, we are interested in studying the higher order Laplacian regularization methods proposed in [16] and studied in depth in [7]. In

particular, we study the higher order Laplacian problem with soft Poisson constraint, namely, the higher order Poisson learning. We first introduce the basic framework of the problem and some useful preliminary results on the graph Laplacian operator. We then use the preliminary results to prove the existence and uniqueness of solutions for higher order Poisson learning problems through method of variations. We develop algorithms for solving the higher order Poisson problems using spectral cut-offs and proceed on running simulations using the MNIST, Fashion-MNIST and cifar-10 data sets to test accuracy and efficiency of the algorithm under very low label rate. We further compare the performance of the algorithm against the other techniques outlined above for semi-supervised graph-based machine learning. We conclude the paper with a discussion on future possibilities and potential improvement for the higher order Poisson method.

Basic Framework

We start with introducing the basic mathematical framework of semi-supervised machine learning. In either fully supervised machine learning, semi-supervised machine learning or unsupervised machine learning, the frameworks are about the same with minor difference in the number of labeled data given in the learning task. Let $\mathcal{X}$ be the set of datapoints that we are interested in doing classification or regression with. In practice, $\mathcal{X}$ is typically $\mathbf{R}^d$, where d represents the number of features of the data. Let $\mathcal{Y}$ be the set of labels. Similarly, $\mathcal{Y}$ is typically $\mathbf{R}^k$ where k is the number of classes. In particular, for a k-class classification problem, the class labels are normally the standard basis vectors in $\mathbf{R}^k$ (one-hot vectors) $e_1, \cdots, e_k \in \mathcal{Y}$. In fully supervised machine learning, we would be given a training subset $\Gamma := \{x_1, \cdots, x_L\} \subseteq \mathcal{X}$ with corresponding labels given as $g : \Gamma \to \mathcal{Y}$. The goal of the fully supervised machine learning algorithm is to find $u : \mathcal{X} \to \mathcal{Y}$ such that $u(x) \approx g(x)$ when $x \in \Gamma$ and propagate the labeling information appropriately for datapoints $\mathcal{X} - \Gamma$. On the other hand, our training set $\Gamma \subseteq \mathcal{X}$ would be completely unlabelled in unsupervised machine learning. We need to use algorithms such as clustering or autoencoders to predict labels

to the data $x \in \mathcal{X}$. In the semi-supervised machine learning setting, we are given an amount of unlabeled data $\{x_{L+1}, \cdots, x_{L+T}\}$ for $T \geq 1$ in addition to our training set $\Gamma = \{x_1, \cdots, x_L\} \subseteq \mathcal{X}$. In particular, the training set Γ can be far smaller than the set of unlabelled data $\{x_{L+1}, \cdots, x_{L+T}\}$, that is, $L \ll T$. The goal of an transductive semi-supervised machine learning algorithm is to learn a function $u : \{x_1, \cdots, x_{L+T}\} \to \mathcal{Y}$ such that $u(x) \approx g(x)$ when $x \in \Gamma$ using the unlabeled data $\{x_{L+1}, \cdots, x_{L+T}\}$ to aid learning.

A common way to develop both unsupervised machine learning and semi-supervised machine learning is to build graphs to encode the similarities between pairs of datapoints using edges. More rigorously, consider $\mathcal{G}(\mathcal{X}, \mathcal{W})$ denoting the graph that has nodes (datapoints) given by $\mathcal{X}$ and the pairwise non-negative weights $(w_{xy})_{x,y \in \mathcal{X}}$ encoding the similarities between vertices $x, y \in \mathcal{X}$. The more similar two vertices are, the larger w_{xy} would be (e.g. $w_{xy} = 1$ when the weights are normalized). In semi-supervised learning, the algorithm is typically found through solving the optimization problem

$$\min_{u:\mathcal{X} \to \mathbf{R^k}} E(u) \text{ Subject To } u(x) \approx g(x) \text{ for } x \in \Gamma,$$

where E is a functional measuring the smoothness for a potential labeling function. The labels $\ell(x)$ for $x \in \mathcal{X} - \Gamma$ are then computed as

$$\ell(x) = \arg\max_{i \in \{1, \cdots, k\}} u_i(x).$$

Next, we give a general review of the graph Laplacian operator and introduce the higher order Poisson learning problem. We largely follow the notation from chapter 5 in [3]. We have a graph $\mathcal{G} = (\mathcal{X}, \mathcal{W})$ where $\mathcal{X}$ are the nodes and $\mathcal{W} = (w_{xy})_{x,y \in \mathcal{X}}$ are the edge weights. Note that the weights are always non-negative.

The graph Laplacian is given by

$$\mathcal{L}u(x) = -\text{div}(\nabla u)(x) = \sum_{y \in \mathcal{X}} w_{xy}(u(x) - u(y)) \tag{2.1}$$

for $u : \mathcal{X} \to \mathbf{R}^k$. Note that the definition of the graph divergence is

$$\text{div}V(x) = \sum_{y \in \mathcal{X}} w_{x,y} V(x,y),$$

for any given vector field $V : \mathcal{X}^2 \to \mathbf{R}^k$. Moreover, the definition of the gradient of u is the vector field

$$\nabla u(x,y) = u(y) - u(x).$$

Note that the graph divergence is defined so that

$$(\nabla u, V)_{\ell^2(\mathcal{X}^2)} = -(u, \text{div}V)_{\ell^2(\mathcal{X})}.$$

The inner product defined for vector fields is

$$(V, W)_{\ell^2(\mathcal{X}^2)} = \frac{1}{2} \sum_{x,y \in \mathcal{X}} w_{xy} V(x,y) W(x,y).$$

Moreover, the inner product for functions $u, v : \mathcal{X} \to \mathbf{R}^k$ is defined as

$$(u, v)_{\ell^2(\mathcal{X})} = \sum_{x \in \mathcal{X}} u(x) \cdot v(x).$$

One can easily show from the above definition that the graph Laplacian is self-adjoint, that is for $u, v : \mathcal{X} \to \mathbf{R}^k$ we have

$$(\mathcal{L}u, v)_{\ell^2(\mathcal{X})} = (u, \mathcal{L}v)_{\ell^2(\mathcal{X})}. \tag{2.2}$$

To see this, note that we have

$$(\mathcal{L}u, v)_{\ell^2(\mathcal{X})} = -(\text{div}(\nabla u), v)_{\ell^2(\mathcal{X})} = (\nabla u, \nabla v)_{\ell^2(\mathcal{X}^2)} = (u, -\text{div}(\nabla v))_{\ell^2(\mathcal{X})} = (u, \mathcal{L}v)_{\ell^2(\mathcal{X})}.$$

The graph Laplacian is also positive semi-definite, that is

$$(u, \mathcal{L}u)_{\ell^2(\mathcal{X})} \geq 0. \tag{2.3}$$

To see this, note that

$$(u, \mathcal{L}u)_{\ell^2(\mathcal{X})} = (\nabla u, \nabla u)_{\ell^2(\mathcal{X}^2)} \geq 0.$$

Laplacian learning extends the labels to the whole graph by solving the optimization problem

$$\min_{u:\mathcal{X}\to\mathbf{R}^k} (u, \mathcal{L}u)_{\ell^2(\mathcal{X})} \text{ Subject To } u(x) = g(x) \text{ for } x \in \Gamma. \tag{2.4}$$

Note that from the definition of graph divergence, we can write our objective function in (2.4) as

$$(u, \mathcal{L}u)_{\ell^2(\mathcal{X})} = (\nabla u, \nabla u)_{\ell^2(\mathcal{X}^2)} = \frac{1}{2}\sum_{x,y} w_{xy}|u(x) - u(y)|^2.$$

We can view the optimization problem (2.4) as looking for a label function u whose labels are the same, or very close, when the weights w_{xy} are large (i.e., x and y are similar). Thus, we are trying to assign the same labels to similar datapoints.

As is shown in [3], the solution of (2.4) coincides with the solution of the Laplace learning equation

$$\begin{cases} \mathcal{L}u(x) = 0 & \text{if } x \in \mathcal{X} \setminus \Gamma \\ u(x) = g(x) & \text{if } x \in \Gamma, \end{cases} \tag{2.5}$$

provided the graph is connected. Note that we shall write the inner products without the subscripts for simplicity. We now briefly recall the argument, since it will be useful for the higher order methods. If u minimizes (2.4) and $v : \mathcal{X} \to \mathbf{R}^k$ is any other function satisfying $v(x) = 0$ for $x \in \Gamma$, then

$$(u, \mathcal{L}u) \leq (u + tv, \mathcal{L}(u + tv))$$

for any number t. This is because $u + tv = g$ on Γ (since $u = g$ and $v = 0$ on Γ), and so $u + tv$ is

a feasible function for the optimization problem and u attains the minimal value. Therefore, the function

$$f(t) = (u + tv, \mathcal{L}(u + tv))$$

attains its minimum at $t = 0$ and so $f'(0) = 0$. Let us write out

$$f(t) = (u, \mathcal{L}u) + [(u, \mathcal{L}v) + (v, \mathcal{L}u)]t + (v, \mathcal{L}v)t^2.$$

Then

$$f'(t) = (u, \mathcal{L}v) + (v, \mathcal{L}u) + 2t(v, \mathcal{L}v),$$

and so

$$f'(0) = (u, \mathcal{L}v) + (v, \mathcal{L}u).$$

Since $\mathcal{L}$ is self-adjoint we have $(u, \mathcal{L}v) = (\mathcal{L}u, v) = (v, \mathcal{L}u)$ and so

$$0 = f'(0) = 2(\mathcal{L}u, v).$$

Since v is arbitrary away from Γ, we find that $\mathcal{L}u(x) = 0$ for $x \in \mathcal{X} \setminus \Gamma$, which establishes (2.5).

One should think of (2.5) as the optimality conditions, that is, the first order necessary conditions $f'(t) = 0$) for a function attaining its minimum value.

The higher order Laplacian methods correspond to the optimization problem

$$\min_{u:\mathcal{X}\to\mathbf{R}^k} (u, \mathcal{L}^m u) \;\text{ Subject To }\; u(x) = g(x) \text{ for } x \in \Gamma, \tag{2.6}$$

where $m \geq 1$ is an integer and is the *order* of the higher order Laplacian method. Increasing m forces the learned function u to be smoother, and is supposed to give better performance at low label rates (but this has not been sufficiently tested).

We can also impose soft label constraints, which can be useful if the labels are noisy, by solving

the problem

$$\min_{u:\mathcal{X}\to\mathbf{R}^k} (u, \mathcal{L}^m u) + \lambda \sum_{x\in\Gamma} \ell(u(x), g(x)), \tag{2.7}$$

where ℓ is a loss function. These are called *soft constraints* because they allow $u(x) \neq g(x)$ for $x \in \Gamma$, whereas (2.6) are referred to as *hard constraints*. It is common to use an L^p loss

$$\ell(u, g) = |u - g|^p.$$

However, inspired by [4], our paper will focus on the more interesting case that uses the Poisson learning loss

$$\ell(u, g) = (\overline{g} - g) \cdot u, \tag{2.8}$$

where $\overline{g} = \frac{1}{|\Gamma|} \sum_{x\in\Gamma} g(x)$ is the average label vector, and $|\Gamma|$ refers to the number of points in Γ. One should note that there is an intriguing similarity between the Poisson loss and the negative log likelihood loss in composition with the softmax function

$$l(u, g) = -g \cdot \log(p),$$

where

$$(p(x))_j := \frac{e^{(u(x))_j}}{\sum_{q=1}^k e^{(u(x))_q}}$$

for $x \in \mathcal{X}$ and $j \in \{1, \cdots, k\}$ being the j-th element of the corresponding vector. In particular, we note that an inner product between the true labels with the labeling function u appears in both of the loss function, except that the true labels are not centered in the negative log likelihood loss composed with softmax comparing to the Poisson learning loss, and the predicted labels in the two

loss functions are normalized differently.

Preliminary Results

We shall proceed to prove a couple of useful lemmas that will be used in the proof of the main theorem.

Lemma 1. *Suppose that the graph Laplacian operator $\mathcal{L}$ is defined on a connected graph $\mathcal{G}$. The kernel of the graph Laplacian operator is the set of constant function on $\mathcal{X}$. In other words, we have*

$$\ker(\mathcal{L}) = \{v \in \ell^2(\mathcal{X}) : v(x) = v(y) \textit{ for all } x, y \in \mathcal{X}\}.$$

Proof. To see this, we let v be a constant function on $\mathcal{X}$. Then we must have

$$\mathcal{L}v(x) = -div(\nabla v)(x) = -\sum_{y \in \mathcal{X}} w_{xy}(v(y) - v(x)) = 0,$$

where the last equality follows from the fact that u is constant on $\mathcal{X}$. Therefore, we have $v \in \ker \mathcal{L}$. Now conversely, suppose that we have $v \in \ker \mathcal{L}$. That is, we have $\mathcal{L}v = 0$, where $v \in \ell^2(\mathcal{X})$. Then it must be the case that

$$\mathcal{L}v(x) = -\sum_{y \in \mathcal{X}} w_{xy}(v(y) - v(x)) = 0$$

for any given $x \in \mathcal{X}$. In particular, if we let $x_0 \in \mathcal{X}$ be the node where v attains its maximum value over $\mathcal{X}$, then we must have

$$0 = \mathcal{L}v(x_0) = -\sum_{y \in \mathcal{X}} w_{x_0 y}(v(y) - v(x_0)) \geq 0.$$

Now note that since we have our graph $\mathcal{G}$ connected, we must have v to be a constant function. To see this, we note that we have, by the connectivity, a set of nodes $\{x^1, ..., x^s\} \subseteq \mathcal{X}$ such that

$w_{x^i,x_0} > 0$ for all $i \in \{1, ..., s\}$. This means that we must have $v(x^i) = v(x_0)$ for all $i \in \{0, ..., s\}$. In particular, this means that we have v attains maximum at $x^i \in \mathcal{X}$ for $i \in \{1, ..., s\}$. Now we have

$$0 = \mathcal{L}v(x^i) = -\sum_{y \in \mathcal{X}} w_{x^i y}(v(y) - v(x^i)) \geq 0$$

for all $i \in \{1, ..., s\}$. For the same reason above, we can see all nodes in $\mathcal{X}$ that connects to x^i will always obtain maximum for v on $\mathcal{X}$. Note that due to the connectivity of the graph, given any point x' x'' in $\mathcal{X}$, we can always find a path that connects x' and x'' such that every node on the path will obtain maximum in v on $\mathcal{X}$. Thus we must have $v \in \ell^2(\mathcal{X})$ to be a constant function. Therefore, we indeed have

$$\ker(\mathcal{L}) = \{v \in \ell^2(\mathcal{X}) : u(x) = u(y) \text{ for all } x, y \in \mathcal{X}\}.$$

□

Lemma 2. *The range of the graph Laplacian operator is the set of unweighted mean-zero functions on $\mathcal{X}$. In other words, we have*

$$\text{range}(\mathcal{L}) = \mathcal{L}(\mathcal{X}) = \{v \in \ell^2(\mathcal{X}) : \sum_{i=1}^{n} v(x_i) = 0\},$$

where $\mathcal{L}$ is the graph Laplacian operator and $\mathcal{X}$ is explicitly written out as $\mathcal{X} := \{x_1, \cdots, x_n\}$ with $n \in \mathbf{Z}^+$.

Proof. We note that $|\text{range}(\mathcal{L})| = |\mathcal{X}| - 1$ by the rank-nullity theorem as $|\ker(\mathcal{L})| = 1$, which is proven in Lemma 1. Therefore, we only need to prove

$$\text{range}(\mathcal{L}) \subseteq \{v \in \ell^2(\mathcal{X}) : \sum_{i=1}^{n} v(x_i) = 0\}$$

as the right hand side has dimension $|\mathcal{X}| - 1$. We can see this by noticing that all functions in $\ell^2(\mathcal{X})$ can be written as the sum of a function of mean zero and the mean of the function which is a constant function. Therefore, we indeed have the right hand side to possess dimension of $|\mathcal{X}| - 1$,

which is the dimension of the set of all functions with finite domain $\mathcal{X}$ minus the dimension of the set of constant functions. Now to prove this inclusion, we let $v \in \ell^2(\mathcal{X})$ and compute

$$\begin{aligned}\sum_{i=1}^{n} \mathcal{L}v(x_i) &= \sum_{i=1}^{n}\sum_{j=1}^{n} w_{x_i x_j}(v(x_i) - v(x_j)) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{n} w_{x_i x_j}(v(x_j) - v(x_i)) \\ &= -\sum_{j=1}^{n}\sum_{i=1}^{n} w_{x_i x_j}(v(x_j) - v(x_i)) \\ &= -\sum_{j=1}^{n} \mathcal{L}v(x_j) \\ &= -\sum_{i=1}^{n} \mathcal{L}v(x_i).\end{aligned}$$

Note that this means that we must have $\sum_{i=1}^{n} \mathcal{L}v(x_i) = 0$. Thus we must have

$$\text{range}\,(\mathcal{L}) \subseteq \{v \in \ell^2(\mathcal{X}) : \sum_{i=1}^{n} v(x_i) = 0\}.$$

Therefore, we have proven that

$$\text{range}\,(\mathcal{L}) = \{v \in \ell^2(\mathcal{X}) : \sum_{i=1}^{n} v(x_i) = 0\}$$

by the discussion at the beginning of the proof. □

Lemma 3. *Let $\mathcal{L}$ be the graph Laplacian operator, $\mathcal{X}$ be explicitly written as $\mathcal{X} := \{x_1, \cdots, x_n\}$ with $n \in \mathbf{Z}^+$, and $m \in \mathbf{Z}^+$ be given. Then we have*

$$\ker\,(\mathcal{L}^m) = \ker\,(\mathcal{L}) = \{v \in \ell^2(\mathcal{X}) : v(x) = v(y) \textit{ for all } x, y \in \mathcal{X}\},$$

and

$$\text{range}\,(\mathcal{L}^m) = \text{range}\,(\mathcal{L}) = \{v \in \ell^2(\mathcal{X}) : \sum_{i=1}^{n} v(x_i) = 0\}.$$

In other words, the kernel and range of the higher order Laplacian is the same as the kernel and range of the graph Laplacian.

Proof. We note that we have proven the last equality for both chains of equality by Lemma 1 and 2. Therefore, we shall only show the first equality for both chains of equality. To this end, we shall proceed by induction. In particular, we consider our base case to be $m = 2$. That is, we shall show that $\ker(\mathcal{L}^2) = \ker \mathcal{L}$ and $\operatorname{range}(\mathcal{L}^2) = \operatorname{range}(\mathcal{L})$. We note that $\operatorname{range}(\mathcal{L}^2)$ is a subspace of $\operatorname{range}(\mathcal{L})$. This is immediately obvious when considering $\mathcal{L}^2$ to be the composition $\mathcal{L} \circ \mathcal{L}$. Therefore, if we could show that $\ker(\mathcal{L}^2) = \ker(\mathcal{L})$, then by the rank-nullity theorem, we have $\operatorname{range}(\mathcal{L}^2) = \operatorname{range}(\mathcal{L})$. To write this in more detail, note that if we have

$$\ker(\mathcal{L}^2) = \ker(\mathcal{L}),$$

then we must have

$$|\ker(\mathcal{L}^2)| = |\mathcal{X}| - |\operatorname{range}(\mathcal{L}^2)| = |\mathcal{X}| - |\operatorname{range}(\mathcal{L})| = |\ker(\mathcal{L})|.$$

This implies that we have $|\operatorname{range}(\mathcal{L}^2)| = |\operatorname{range}(\mathcal{L})|$ and $\operatorname{range}(\mathcal{L}^2)$ is a subspace of $\operatorname{range}(\mathcal{L})$. Therefore, we must have $\operatorname{range}(\mathcal{L}^2) = \operatorname{range}(\mathcal{L})$. According to the above logic, now we shall proceed to show

$$\ker(\mathcal{L}^2) = \ker(\mathcal{L}).$$

To this end, we let $u \in \ell^2(\mathcal{X})$ such that $\mathcal{L}^2 u = 0$. Therefore, we have

$$\mathcal{L}(\mathcal{L}u) = 0.$$

This means that $\mathcal{L}u \in \ker(\mathcal{L})$. However, we know that the kernel of $\mathcal{L}$ is the set of constant functions on $\mathcal{X}$. This mean that we have

$$\mathcal{L}u = c,$$

for some $c \in \mathbf{R}$. Note further that $\mathcal{L}u \in \text{range}\,(\mathcal{L})$ by definition. Therefore, we must have $\mathcal{L}u$ to be of mean zero. In particular, this means that

$$\mathcal{L}u = c$$

and

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

Therefore, we must have $\mathcal{L}u(x) = 0$ for all $x \in \mathcal{X}$. That is, we have $\mathcal{L}u \equiv 0$. Therefore, we must have $u \in \ker(\mathcal{L})$ by definition. Now conversely, we have $\ker(\mathcal{L}) \subseteq \ker(\mathcal{L}^2)$ to hold trivially by the fact that $\mathcal{L}$ is a linear operator. Therefore, we have shown that

$$\ker(\mathcal{L}^2) \subseteq \ker(\mathcal{L}).$$

Now suppose inductively that

$$\ker(\mathcal{L}^m) = \ker(\mathcal{L})$$

for some $m \in \mathbf{Z}^+$. We want to show that

$$\ker(\mathcal{L}^{m+1}) = \ker(\mathcal{L}).$$

Note that proving the right hand side if contained in the left hand side is trivial due to the fact that $\mathcal{L}$ is a linear operator. To show that the left hand side is contained in the right, we shall apply a similar logic from the base case proof. To this end, we let $u \in \ell^2(\mathcal{X})$ be such that

$$\mathcal{L}^{m+1}u = 0.$$

this means that we have

$$\mathcal{L}^m(\mathcal{L}u) = 0.$$

Now this means that

$$\mathcal{L}u \in \ker\left(\mathcal{L}^m\right) = \ker\left(\mathcal{L}\right).$$

However, this implies that we have

$$\mathcal{L}u = c',$$

for some $c' \in \mathbf{R}$. Moreover, we have $\mathcal{L}u \in \operatorname{range}\left(\mathcal{L}\right)$. This means that we have that $\mathcal{L}u$ is a mean zero function. Therefore, we must have $\mathcal{L}u = 0$ by the same logic when proving the base case. Thus we have $u \in \ker\left(\mathcal{L}\right)$. Therefore, we can conclude that, by the same logic as in the base case, that

$$|\operatorname{range}\left(\mathcal{L}^{m+1}\right)| = |\operatorname{range}\left(\mathcal{L}\right)|.$$

This closes the induction. Therefore, we have proven the desired result by the discussion at the beginning of the proof. □

The Proof of Existence and Uniqueness of the Solution

With all of the lemmas proven in the previous chapter, we shall now proceed to state and prove the existence and uniqueness of the solution for higher order Poisson learning problems though variational analysis. Note that using the Poisson learning loss for (2.7) corresponds to the variational problem

$$\min_{u \in \ell_0^2(\mathcal{X})} \left\{ \frac{1}{2}(u, \mathcal{L}^m u) - \sum_{x \in \Gamma} (g(x) - \overline{g}) \cdot u(x) \right\}, \tag{4.1}$$

where $\overline{g} = \frac{1}{|\Gamma|} \sum_{x \in \Gamma} g(x)$ is the average label vector. Here, $\ell_0^2(\mathcal{X})$ is the space of functions $u : \mathcal{X} \to \mathbf{R}^k$ that have mean-value zero; that is

$$\ell_0^2(\mathcal{X}) = \left\{ u : \mathcal{X} \to \mathbf{R}^k \, : \, \sum_x d(x) u(x) = 0 \right\},$$

where $d(x) = \sum_{y \in \mathcal{X}} w_{xy}$ is the degree. The Euler-Lagrange equation satisfied by minimizers should be

$$\mathcal{L}^m u(x) = \sum_{y \in \Gamma} (g(y) - \overline{g}) \delta_{xy}, \tag{4.2}$$

which is the m^{th} order Poisson equation. Note that δ_{xy} is the Kronecker delta, satisfying $\delta_{xy} = 1$ when $x = y$ and $\delta_{xy} = 0$ otherwise. We shall first show that the existence and uniqueness of solution for the Euler-Lagrange equation (4.2). One should note that we are restricting the solution space to the mean-zero function space $\ell_0^2(\mathcal{X})$ due to the fact that the kernel of $\mathcal{L}^m$ is non-trivial and one-dimensional as proven in Lemma 3. In particular, in order for us to obtain a unique solution for (4.2), we shall impose an additional constraint by requiring the weighted mean of $u \in \ell^2(\mathcal{X})$ to be zero. The specific choice of weight by degrees comes naturally from the random walk interpretation of Poisson learning in [4].

Theorem 4. *Let $\mathcal{L}$ be the graph Laplacian operator and $m \in \mathbf{Z}^+$ be given. The Euler Lagrange equation*

$$\mathcal{L}^m u(x) = \sum_{y \in \Gamma} (g(y) - \overline{g}) \delta_{xy},$$

has a unique solution $u \in \ell_0^2(\mathcal{X})$. Note that $g \in \ell^2(\Gamma)$ is the labeling function at the boundary points, whereas $\overline{g} = \frac{1}{|\Gamma|} \sum_{x \in \Gamma} g(x)$ is the average label vector and $|\Gamma|$ represents the number of points in Γ.

Proof. We shall first note that

$$\mathcal{L}^m u(x) = \begin{cases} g(x) - \overline{g}, & \text{when } x \in \Gamma, \\ 0, & \text{when } x \notin \Gamma. \end{cases}$$

In particular, we will always have an unweighted mean-zero function on the right hand side. Recall that we proved that $\text{range}(\mathcal{L}^m)$ is the set of all unweighted mean-zero functions in Lemma 3. Therefore, there must exist some $u \in \ell^2(\mathcal{X})$ such that $\mathcal{L}^m u(x) = \sum_{y \in \Gamma} (g(y) - \overline{g}) \delta_{xy}$. Moreover, this solution is unique up to a constant function. To see this, let $u, v \in \ell^2(\mathcal{X})$ be given such that

both u and v solves equation (4.2). In other words, we have $u, v \in \ell^2(\mathcal{X})$ such that

$$\mathcal{L}^m u = \sum_{y \in \Gamma} (g(y) - \overline{g}) \delta_{xy} = \mathcal{L}^m v.$$

Thus we must have

$$\mathcal{L}^m u - \mathcal{L}^m v = 0.$$

In particular, we have, by linearity, that

$$\mathcal{L}^m (u - v) = 0.$$

However, this means that $u - v \in \ker(\mathcal{L}^m)$ by Lemma 3. Thus we have

$$u - v = c,$$

where c is some constant function by the characteristic of the kernel of $\mathcal{L}^m$. Equivalently, we have $u = c + v$. Thus we have our solution u to be the unique up to a constant. Therefore, we have proven the desired result. □

With Theorem 4 in hands, we prove that the variational problem (4.1) admits a unique minimizer up to a constant function.

Theorem 5. *The variational problem for the higher order Poisson regularization*

$$\min_{u \in \ell_0^2(\mathcal{X})} \left\{ \frac{1}{2}(u, \mathcal{L}^m u) - \sum_{x \in \Gamma} (g(x) - \overline{g}) \cdot u(x) \right\},$$

admits a unique minimizer.

Proof. Take u solving (4.2). Let $v \in \ell_0^2(\mathcal{X})$ be any other function with zero mean value. We just

need to show that $I(u) \leq I(v)$, where

$$I(u) := \frac{1}{2}(u, \mathcal{L}^m u) - \sum_{x \in \Gamma} (g(x) - \overline{g}) \cdot u(x).$$

This we can prove fairly directly. Let us write

$$f(x) = \sum_{y \in \Gamma} (g(y) - \overline{g}) \delta_{xy}$$

so that

$$I(u) = \frac{1}{2}(u, \mathcal{L}^m u) - \sum_{x \in \mathcal{X}} f(x) \cdot u(x),$$

and (4.2) becomes $\mathcal{L}^m u = f$. We will show that

$$\frac{d}{dt} I(u + t(v - u)) \geq 0 \tag{4.3}$$

for $t > 0$, from which we can deduce

$$I(v) - I(u) = \int_0^1 \frac{d}{dt} I(u + t(v - u))\, dt \geq 0,$$

and so $I(v) \geq I(u)$. To see this, we compute

$$\begin{aligned}
\frac{d}{dt} I(u + t(v - u)) &= \frac{d}{dt} \left[\frac{1}{2}(u + t(v - u), \mathcal{L}^m(u + t(v - u))) - (u + t(v - u), f) \right] \\
&= \frac{1}{2}(v - u, \mathcal{L}^m(u + t(v - u))) + \frac{1}{2}(u + t(v - u), \mathcal{L}^m(v - u)) - (v - u, f) \\
&= t(v - u, \mathcal{L}^m(v - u)) + (v - u, \mathcal{L}^m u) - (v - u, f) \\
&= t(v - u, \mathcal{L}^m(v - u)),
\end{aligned}$$

where we used $\mathcal{L}^m u = f$ in the last line, and self-adjointness of $\mathcal{L}^m$ in the previous line. Since $\mathcal{L}$ is positive definite, we obtain (4.3), which establishes that $I(u) \leq I(v)$.

Note we did not use that $v \in \ell_0^2(\mathcal{X})$; i.e., it did not matter that v had mean-zero. The mean-zero

condition is only used to select the unique solution, up to a constant function. □

This conclude the proof of Existence and Uniqueness of the solution for higher order Poisson learning problems.

Algorithm Development and Simulations

We develop an algorithm to solve higher order Poisson problems. One should note that we need to iteratively multiply the graph Laplacian matrix to construct a higher order Laplacian operator. Moreover, solving the Euler-Lagrange equation (4.2) for the first order Poisson learning problem is equivalent to solving a linear system $Ax = b$. The computational complexity of solving such a system depends on the condition numbers of A. Larger condition numbers indicate ill-conditioned systems that are harder for the computers to solve. The condition number of A^2 is the square of the condition number. Similarly, the condition number of A^m is the m^{th} power of the condition number of A. Therefore, the problem will become harder to solve when m gets larger. To make the computational complexity more tractable, we will approximate the solution through spectral numerical methods.

Note that we shall proceed with one caveat in mind. The fact that the linear system that we are interested in solving has a high conditional number also implies that the accuracy of the solution might be low, unless our right hand side b in $Ax = b$ is chosen to be an eigenvector of A. Such loss of accuracy in the solution is independent of the algorithm, so one must proceed with caution.

5.1 Spectral Numerical Methods

We shall give a brief introduction to spectral numerical methods that we will be using to approximate the solution to the higher order Poisson learning problems. In particular, we outline

here how to solve the Poisson equation

$$\mathbf{L}^m \mathbf{x} = \mathbf{F} \tag{5.1}$$

with a spectral approximation, where $m > 0$ is any real number. Here $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the unnormalized graph Laplacian for the $n \times n$ symmetric weight matrix $\mathbf{W}$, and $\mathbf{D}$ is the corresponding diagonal degree matrix with entries $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{W}_{ij}$. The right hand side $\mathbf{F}$ is an $n \times k$ matrix, and the solution $\mathbf{x}$ is also an $n \times k$ matrix, making this a collection of k linear equations. Note that $\mathbf{F}$ is the right hand side Poisson source term of (4.2). For solvability, the Poisson source term $\mathbf{F}$ must be mean zero, that is

$$\mathbb{1}^T \mathbf{F} = 0$$

where $\mathbb{1}$ is the $n \times 1$ all ones vector. One should note that this is indeed true as we have

$$\begin{aligned} \sum_{x \in \mathcal{X}} \sum_{y \in \Gamma} (g(y) - \overline{g}) \delta_{xy} &= \sum_{x \in \Gamma} (g(x) - \overline{g}) \\ &= \sum_{x \in \Gamma} g(x) - \sum_{x \in \Gamma} g(x) \\ &= 0. \end{aligned}$$

The graph Laplacian matrix $\mathbf{L}$ is an $n \times n$ positive semi-definite symmetric matrix, and is thus diagonalizable. This means there exists an $n \times n$ orthogonal matrix $\mathbf{V}$ (i.e., $\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$) and an $n \times n$ diagonal matrix $\mathbf{\Lambda}$ such that

$$\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T.$$

The columns of $\mathbf{V}$ are exactly the eigenvectors of $\mathbf{L}$, and the diagonal entries of $\mathbf{\Lambda}$ are the corresponding non-negative eigenvalues. We write $\lambda_i = \mathbf{\Lambda}_{ii}$ for the eigenvalues for convenience. We assume the eigenvalues are ordered from least to greatest, so

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n.$$

The m^{th} power of $\mathbf{L}$ for any real number $m > 0$ is defined as

$$\mathbf{L}^m = \mathbf{V}\mathbf{\Lambda}^m\mathbf{V}^T. \tag{5.2}$$

The definition of $\mathbf{\Lambda}^m$ is to raise the diagonal entries of $\mathbf{\Lambda}$ to the power m, that is

$$\mathbf{\Lambda}^m = \begin{bmatrix} \lambda_1^m & & & \\ & \lambda_2^m & & \\ & & \ddots & \\ & & & \lambda_n^m. \end{bmatrix}$$

This agrees with usual notion of the m^{th} power

$$\mathbf{L}^m = \underbrace{\mathbf{L}\mathbf{L}\cdots\mathbf{L}}_{\text{m times}}$$

when m is a positive integer. Indeed, we have

$$\mathbf{L}^2 = \mathbf{L}\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{V}\mathbf{\Lambda}^2\mathbf{V}^T$$

since $\mathbf{V}$ is orthogonal. We can continue by induction for higher powers. The advantage of the spectral definition of power (5.2) is that it applies to any real number $m > 0$, and it is trivial to compute $\mathbf{\Lambda}^m$ since $\mathbf{\Lambda}$ is a diagonal matrix.

To solve (5.1), we write out the equation with the spectral definition of $\mathbf{L}^m$ as

$$\mathbf{V}\mathbf{\Lambda}^m\mathbf{V}^T\mathbf{x} = \mathbf{F}.$$

We can multiply by $\mathbf{V}^T$ on both sides to get

$$\mathbf{\Lambda}^m\mathbf{V}^T\mathbf{x} = \mathbf{V}^T\mathbf{F}.$$

Since $\mathbf{\Lambda}^m$ is a diagonal matrix, its inverse is simple to compute by inverting the diagonal entries. The only issue is that $\lambda_1 = 0$. We assume the graph is connected, in which λ_1 is the only null eigenvalue, and the corresponding eigenvector is the all ones vector $\mathbb{1}$. This is where the mean-zero condition comes in. If we look for a solution $\mathbf{x}$ with mean-value zero, then we can ignore λ_1. Hence, we can use the pseudo-inverse of $\mathbf{\Lambda}^m$, which involves inverting only the nonzero diagonal entries. We write this as

$$\mathbf{\Lambda}_m := \begin{bmatrix} 0 & & & \\ & \lambda_2^{-m} & & \\ & & \ddots & \\ & & & \lambda_n^{-m}. \end{bmatrix}$$

This suggests the solution formula

$$\mathbf{x} = \mathbf{V}\mathbf{\Lambda}_m\mathbf{V}^T\mathbf{F} \tag{5.3}$$

for the solution of (5.1). To check that the solution formula works, we compute

$$\mathbf{L}^m\mathbf{x} = \mathbf{V}\mathbf{\Lambda}^m\mathbf{V}^T\mathbf{x} = \mathbf{V}\mathbf{\Lambda}^m\mathbf{V}^T\mathbf{V}\mathbf{\Lambda}_m\mathbf{V}^T\mathbf{F} = \mathbf{V}(\mathbf{\Lambda}^m\mathbf{\Lambda}_m)\mathbf{V}^T\mathbf{F}.$$

The matrix $\mathbf{\Lambda}^m\mathbf{\Lambda}_m$ is nearly the identity, except for the first entry, that is

$$\mathbf{\Lambda}^m\mathbf{\Lambda}_m = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1. \end{bmatrix}$$

Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ denote the columns of $\mathbf{V}$ (i.e., the eigenvectors of $\mathbf{L}$). The right hand side can be

computed as

$$\begin{aligned}\mathbf{V}(\mathbf{\Lambda}^m\mathbf{\Lambda}_m)\mathbf{V}^T &= \begin{bmatrix}\mathbf{v}_1\,\mathbf{v}_2\,\cdots\,\mathbf{v}_n\end{bmatrix}\begin{bmatrix}0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1.\end{bmatrix}\begin{bmatrix}\mathbf{v}_1^T\\ \mathbf{v}_2^T\\ \vdots\\ \mathbf{v}_n^T\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{0}\,\mathbf{v}_2\,\cdots\,\mathbf{v}_n\end{bmatrix}\begin{bmatrix}\mathbf{v}_1^T\\ \mathbf{v}_2^T\\ \vdots\\ \mathbf{v}_n^T\end{bmatrix}\\ &= \mathbf{v}_2\mathbf{v}_2^T + \mathbf{v}_3\mathbf{v}_3^T + \cdots + \mathbf{v}_n\mathbf{v}_n^T.\end{aligned}$$

We also clearly have

$$\mathbf{V}\mathbf{V}^T = \mathbf{v}_1\mathbf{v}_1^T + \mathbf{v}_2\mathbf{v}_2^T + \mathbf{v}_3\mathbf{v}_3^T + \cdots + \mathbf{v}_n\mathbf{v}_n^T.$$

Since the first eigenvector is $\mathbf{v}_1 = \frac{1}{\sqrt{n}}\mathbb{1}$, and $\mathbb{1}^T\mathbf{F} = 0$ (the mean-zero condition), we have that $\mathbf{v}_1^T\mathbf{F} = 0$, and so

$$\mathbf{V}(\mathbf{\Lambda}^m\mathbf{\Lambda}_m)\mathbf{V}^T\mathbf{F} = \mathbf{V}\mathbf{V}^T\mathbf{F} = \mathbf{F}.$$

This shows that $\mathbf{L}^m\mathbf{x} = \mathbf{F}$, and so the formula (5.3) indeed solves (5.1) whenever $\mathbf{F}$ satisfies the mean-zero condition $\mathbb{1}^T\mathbf{F} = 0$.

To check that (5.3) gives a solution $\mathbf{x}$ with mean-value zero, we compute

$$\frac{1}{\sqrt{n}}\mathbb{1}^T\mathbf{x} = \mathbf{v}_1^T\mathbf{x} = \mathbf{v}_1^T\mathbf{V}\mathbf{\Lambda}_m\mathbf{V}^T\mathbf{F} = \mathbf{e}_1^T\mathbf{\Lambda}_m\mathbf{V}^T\mathbf{F} = \mathbf{0},$$

where $\mathbf{e}_1 = [1, 0, \ldots, 0]^T$ and we used that the columns of $\mathbf{V}$ are mutually orthogonal (since $\mathbf{V}$ is an orthogonal matrix).

The solution formula (5.3) is still intractable, because we cannot compute all n eigenvectors of a large matrix, simply due to lack of computation time and memory storage. However, we can

now introduce a spectral approximation. Let $\mathbf{V}_N$ denote the $n \times N$ matrix consisting of the first N columns of $\mathbf{V}$, that is

$$\mathbf{V}_N = \begin{bmatrix} \mathbf{v}_1 \, \mathbf{v}_2 \, \cdots \mathbf{v}_N \end{bmatrix},$$

where $N \ll n$. A spectral cutoff approximation of the solution formula (5.3) is given by

$$\mathbf{x}_N = \mathbf{V}_N \mathbf{\Lambda}_{N,m} \mathbf{V}_N^T \mathbf{F}, \tag{5.4}$$

where $\mathbf{\Lambda}_{N,m}$ is the $N \times N$ diagonal matrix given by

$$\mathbf{\Lambda}_{N,m} := \begin{bmatrix} 0 & & & \\ & \lambda_2^{-m} & & \\ & & \ddots & \\ & & & \lambda_N^{-m}. \end{bmatrix}$$

If we take a reasonable value for N, the approximate solution formula (5.4) is tractable, since we only need to compute N eigenvectors, and we can carefully do the computation in (5.4) to avoid constructing a dense $n \times n$ matrix. That is, we do the computation as

$$\mathbf{x}_N = (\mathbf{V}_N \mathbf{\Lambda}_{N,m}) \cdot (\mathbf{V}_N^T \mathbf{F}),$$

which involves working with the $n \times N$ matrix $\mathbf{V}_N \mathbf{\Lambda}_{N,m}$ and the $N \times k$ matrix $\mathbf{V}_N^T \mathbf{F}$.

The spectral approximation does not give the exact solution, but instead describes the smoothly varying part of the solution, and is accurate when there is very little energy content in the solution $\mathbf{x}$ at higher frequencies. Indeed, we can write out both solution formulas to compare them. We have

$$\mathbf{x} = \lambda_2^{-m} \mathbf{v}_2 \mathbf{v}_2^T \mathbf{F} + \lambda_3^{-m} \mathbf{v}_3 \mathbf{v}_3^T \mathbf{F} + \cdots + \lambda_n^{-m} \mathbf{v}_n \mathbf{v}_n^T \mathbf{F},$$

and

$$\mathbf{x}_N = \lambda_2^{-m}\mathbf{v}_2\mathbf{v}_2^T\mathbf{F} + \lambda_3^{-m}\mathbf{v}_3\mathbf{v}_3^T\mathbf{F} + \cdots + \lambda_n^{-m}\mathbf{v}_N\mathbf{v}_N^T\mathbf{F}.$$

Thus, the difference between the two solutions is found only in the higher frequencies (hence the term *spectral cutoff*) and we have

$$\mathbf{x} - \mathbf{x}_N = \lambda_{N+1}^{-m}\mathbf{v}_2\mathbf{v}_2^T\mathbf{F} + \lambda_{N+2}^{-m}\mathbf{v}_3\mathbf{v}_3^T\mathbf{F} + \cdots + \lambda_n^{-m}\mathbf{v}_n\mathbf{v}_n^T\mathbf{F}$$

and so

$$\|\mathbf{x} - \mathbf{x}_N\| = \lambda_{N+1}^{-m}\|\mathbf{v}_2^T\mathbf{F}\| + \lambda_{N+2}^{-m}\|\mathbf{v}_3^T\mathbf{F}\| + \cdots + \lambda_n^{-m}\|\mathbf{v}_n^T\mathbf{F}\|.$$

In general, we expect the eigenvalues λ_i to grow rapidly as $i \to n$, so the error is small, especially when m is large.

5.2 Higher Order Poisson Learning Algorithm

With the spectral numerical approximation method introduced in the previous section, we now provide the pseudocode for the higher order Poisson learning algorithm. To construct the graph Laplacian matrix, we let $\mathbf{W} = (w_{ij})_{i,j=1}^n$ be the symmetric weight matrix, $\mathbf{D} = \text{diag}(d_i)_{i=1}^n$ be the diagonal degree matrix and construct graph Laplacian matrix $\mathbf{L} := W - D$. We denote N as the spectral cutoff, $vals$ as the eigenvalues vector of $\mathbf{L}$, $vecs$ as the eigenvectors matrix of $\mathbf{L}$ with the eigenvectors as the columns, m as the order of the higher order graph Laplacian matrix $\mathbf{L}^m$, and x_N as the solution when the spectral cutoff is N.

Algorithm 5.1: Higher Order Poisson Learning

```
input: N, vals, vecs, m
output: x_N
begin
    b ← the Poisson source term
    vals ← the first N values of vals excluding the first value
    vecs ← the first N columns of vecs excluding the first column
    inverse_vals ← invert the vals and raise each value of the vector to the power of m
```

```
    Lambda ← the diagonal matrix where the values of inverse_vals are the diagonal entries
    x_N ← (vecs multiplies Lambda) multiplies (transpose of vecs multiplies b)
    return x_N
end
```

In particular, we note that the Poisson source term can be constructed through

$$b := \sum_{y \in \Gamma} (g(y) - \overline{g}) \delta_{xy}$$

given in the right hand side of (4.2).

5.3 Simulation Results

We present the best choice of spectral cutoff and order to use for the higher order Poisson algorithm by running the algorithm on the following three datasets: MNIST, Fashion-MNIST and cifar-10, for multi-class classification under very low label rates. Moreover, we compare the performance of the higher order Poisson algorithm with the other semi-supervised machine learning algorithms evaluated in [4].

Note that to safely compare the higher order Poisson algorithm with the ones evaluated in [4], we shall follow the same procedure of graph construction for each data set as in [4]. We run 100 trials by randomly choosing labeled data for all combinations of spectral cutoffs $N \in \{10, 20, 30, 40, 50, 100, 150, 200, 250, 300\}$ and orders $m \in \{0.1, 0.2, 0.3, \cdots, 5.0\}$. The best results are presented here in the paper and the other results are archived in a GitHub repository.[1] For easy comparison with the results presented in [4], we shall include the results of the standard Poisson learning in [4] in our tables as well. Note that we use the result for standard Poisson learning as a reference since the Poisson learning outperforms the other learning algorithms that we wish to compare our algorithm with. We also present the visualization of the results on each dataset when one labeled data per class is given. The other visualizations with higher label rates will be

[1] **GitHub Address**: https://github.com/DingjunB/Honors-Thesis.git

presented in the Appendix.

We observe that our higher order Poisson (HOP) algorithm performs differently on different data sets. However, the algorithm generally performs better when the spectral cutoff is low. We can see this from both the data and the visualization of the accuracy scores. This is a reasonable observation as the first few eigenvectors, like in spectral clustering, describe the cluster structure in the data. The eigenvectors with higher eigenvalues are describing finer details in the graph and do not provide much information about the large clusters.

Table 5.1: MNIST: Mean (standard deviation) classification accuracy over 100 trials.

# LABELS PER CLASS	**1**	**2**	**3**	**4**	**5**
HOP WITH $N = 20$ & $m = 1.0$	**91.6 (3.9)**	**94.3 (1.6)**	**95.0 (0.8)**	95.1 (0.6)	95.3 (0.5)
HOP WITH $N = 30$ & $m = 1.0$	90.9 (3.8)	94.3 (1.5)	94.9 (1.1)	**95.3 (0.9)**	**95.6 (0.8)**
POISSON	90.2 (4.0)	93.6 (1.6)	94.5 (1.1)	94.9 (0.8)	95.3 (0.7)

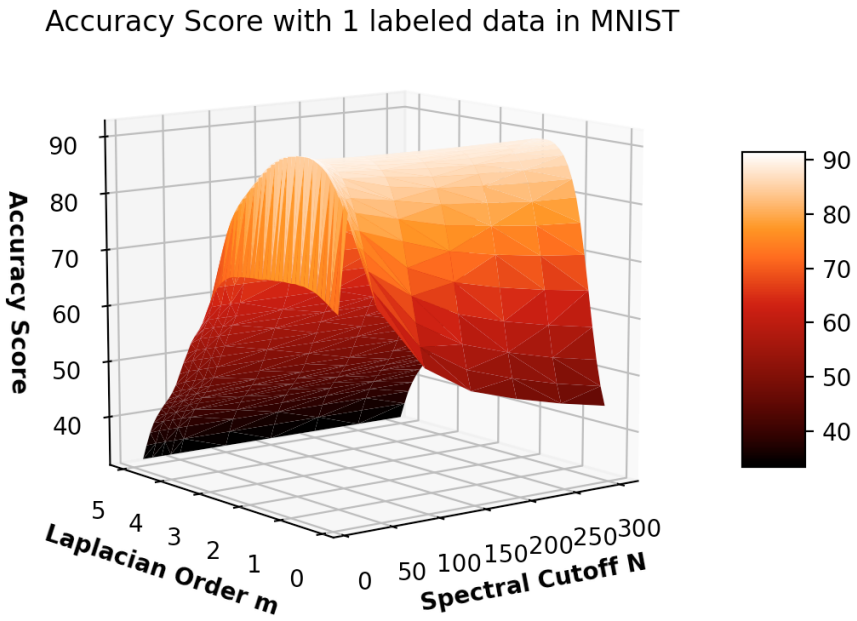

Figure 5.1: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in MNIST.

It is worth noticing from Table 5.1 that our algorithm performs the best when the order is around $m = 1$ for the MNIST data set. In particular, one should note that the higher order Poisson

algorithm would degenerate to the basic Poisson learning algorithm [4] when the order is $m = 1$. However, we have an extra hyper-parameter known as the spectral cutoff N coming from the spectral approximation in the development of our algorithm. This extra parameter provides us a slight advantage in terms of both the accuracy score and the flexibility of the learning algorithm comparing to the standard Poisson learning algorithm.

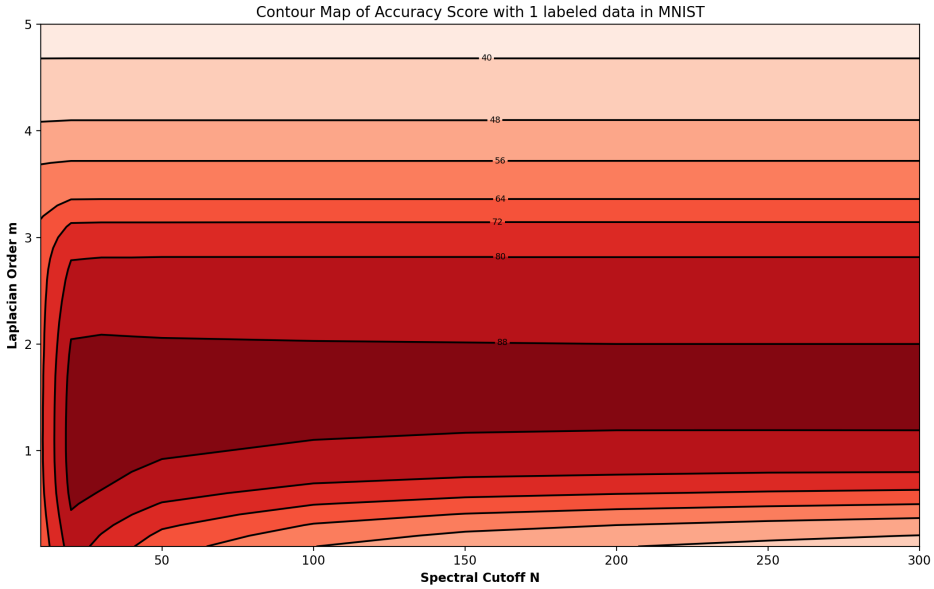

Figure 5.2: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in MNIST.

We observe from Figure 5.1 and 5.2 that the accuracy of the HOP algorithm decreases significantly with the increase of the Laplacian order m. In particular, the accuracy score peaks when the Laplacian order m is between 1 and 2 given any spectral cutoff N. Moreover, we observe from 5.2 that the accuracy score is more likely to decreases with the increase of spectral cutoff N and the accuracy score is more likely to be high when the spectral cutoff N is less than 50 given any order m.

Moving on to the Fashion-MNIST dataset, we observe from Table 5.2 that the accuracy score of the HOP algorithm peaks when m is less than 1. Moreover, Figure 5.3 and 5.4 shows that we have the accuracy score to decrease monotonely after $m \approx 1$ given any spectral cutoff N. On the

other hand, we observe from Figure 5.4 that increase of spectral cutoff N will lead to a decrease in the accuracy rate when $m \leq 1$. We note also from Figure 5.4 that the accuracy is most likely to be maximized when N is less than 50 given any Laplacian order m. Therefore, the accuracy score of the HOP algorithm is generally higher when $0 \leq m \leq 1$ and N is low. Moreover, we note from Table 5.2 that the standard Poisson learning algorithm outperforms HOP in almost all cases except from when the number of labels per class is one. However, the accuracy margin between HOP and the standard Poisson learning is minimal, and HOP can potentially compensate this margin through fine-tuning the parameter value for N and m.

Table 5.2: Fashion-MNIST: Mean (standard deviation) classification accuracy over 100 trials.

# LABELS PER CLASS	**1**	**2**	**3**	**4**	**5**
HOP WITH $N = 10$ & $m = 0.4$	**61.7 (5.2)**	65.6 (4.1)	68.5 (2.6)	69.5 (2.2)	69.7 (2.1)
HOP WITH $N = 20$ & $m = 0.7$	60.7 (4.8)	**65.8 (4.4)**	69.4 (2.8)	70.6 (2.6)	71.4 (2.3)
HOP WITH $N = 30$ & $m = 0.8$	59.9 (4.7)	65.6 (4.2)	**69.5 (2.7)**	70.8 (2.5)	72.0 (2.3)
HOP WITH $N = 30$ & $m = 0.7$	59.6 (4.7)	65.5 (4.2)	69.4 (2.7)	**70.9 (2.5)**	**72.1 (2.2)**
POISSON	60.8 (4.6)	**66.1 (3.9)**	**69.6 (2.6)**	**71.2 (2.2)**	**72.4 (2.3)**

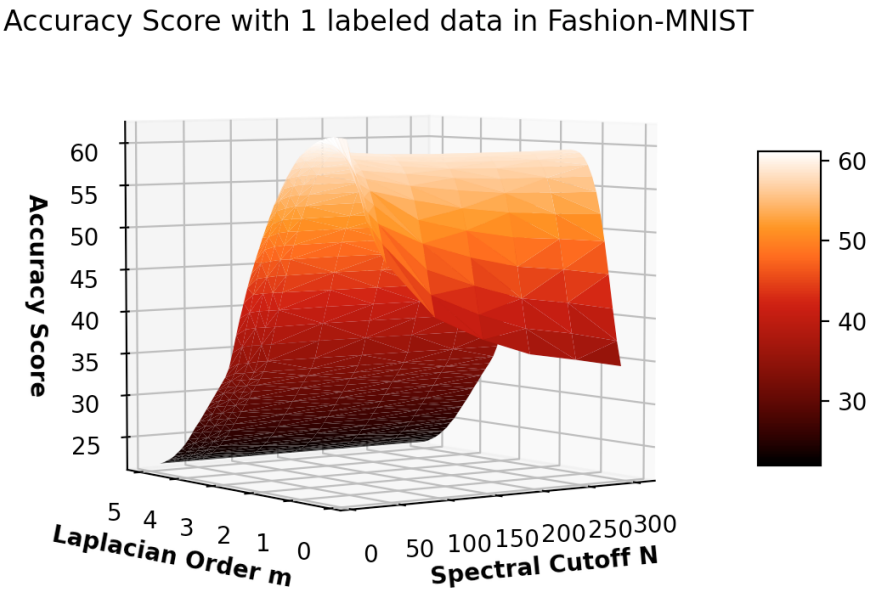

Figure 5.3: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in Fashion-MNIST.

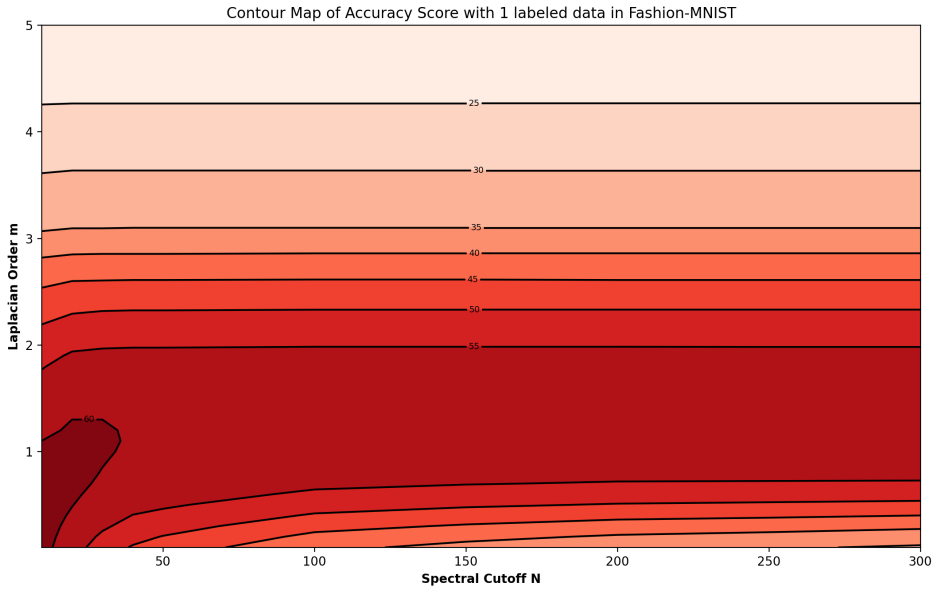

Figure 5.4: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in Fashion-MNIST.

We again observe a similar pattern for the Cifar-10 data set from Table 5.3. The accuracy rate peaks when $N = 20$ and $0 \leq m \leq 1$. Moreover, we observe that the parameter flexibility in our learning algorithm provides us with a slight advantage in accuracy scores when comparing with the standard Poisson learning algorithm. We also note from Figure 5.5 and 5.6 that the high accuracy scores are obtained more often when m is between 0 and 2 when N is small. Furthermore, we observe from Figure 5.6 that the highest accuracy scores happens more often when $1 \leq m \leq 2$ when N is large.

Table 5.3: Cifar-10: Mean (standard deviation) classification accuracy over 100 trials.

# LABELS PER CLASS	**1**	**2**	**3**	**4**	**5**
HOP WITH $N = 20$ & $m = 0.3$	**41.8 (5.5)**	47.7 (5.3)	51.0 (3.9)	53.1 (3.1)	54.4 (2.7)
HOP WITH $N = 20$ & $m = 0.1$	41.7 (5.5)	**47.8 (5.3)**	**51.0 (3.8)**	**53.2 (3.1)**	**54.5 (2.7)**
POISSON	40.7 (5.5)	46.5 (5.1)	49.9 (3.4)	52.3 (3.1)	53.8 (2.6)

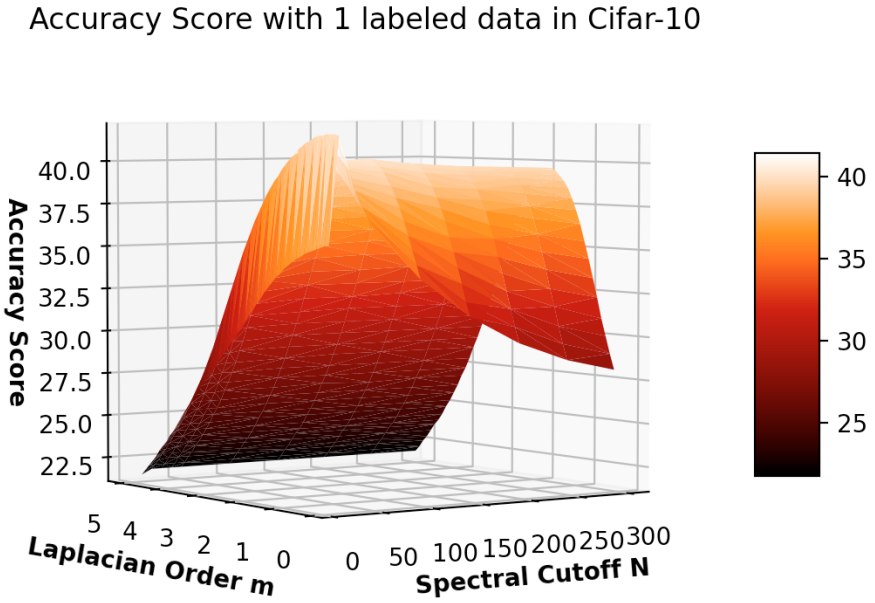

Figure 5.5: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in Cifar-10.

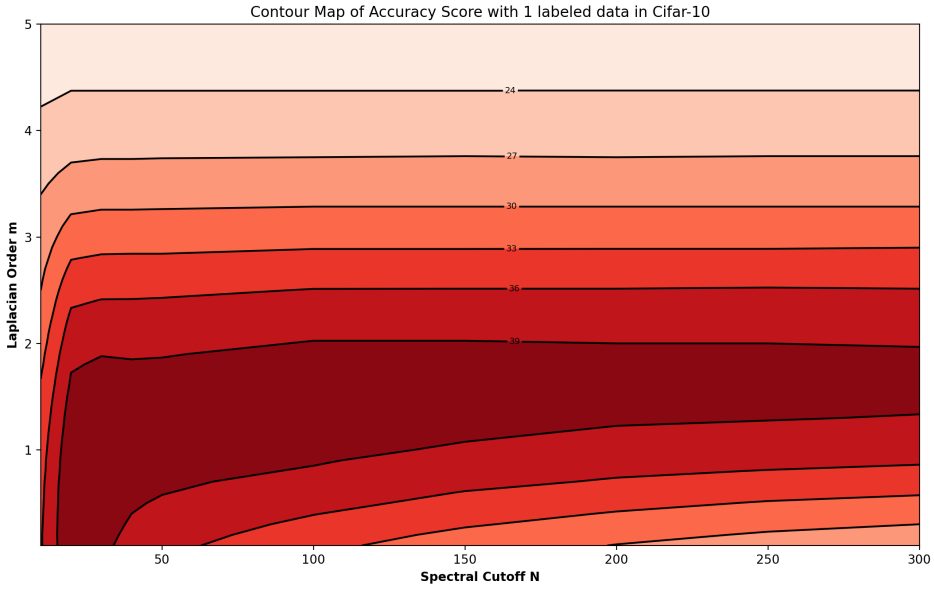

Figure 5.6: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in Cifar-10.

We also observe that by fixing a higher spectral cutoff, we would force the order of the algorithm to become greater or equal to 1 in order to obtain the best performance for all simulations through all data sets. We shall illustrate this observation here using the case when $N = 200$ as an example.

Table 5.4: MNIST: Mean (standard deviation) classification accuracy over 100 trials with $N = 200$.

# LABELS PER CLASS	**1**	**2**	**3**	**4**	**5**
HOP WITH $N = 200$ & $m = 1.6$	**89.3 (4.1)**	92.8 (2.3)	93.6 (2.2)	93.9 (1.9)	94.2 (1.6)
HOP WITH $N = 200$ & $m = 1.4$	89.1 (4.0)	**93.0 (1.9)**	**94.0 (1.6)**	94.4 (1.4)	94.8 (1.2)
HOP WITH $N = 200$ & $m = 1.3$	88.7 (4.0)	92.9 (1.8)	94.0 (1.4)	**94.5 (1.2)**	**94.9 (1.1)**
POISSON	**90.2 (4.0)**	**93.6 (1.6)**	**94.5 (1.1)**	**94.9 (0.8)**	**95.3 (0.7)**

Table 5.5: Fashion-MNIST: Mean (standard deviation) classification accuracy over 100 trials with $N = 200$.

# LABELS PER CLASS	**1**	**2**	**3**	**4**	**5**
HOP WITH $N = 200$ & $m = 1.3$	**59.2 (4.4)**	64.2 (3.8)	67.8 (2.6)	68.9 (2.3)	70.4 (2.5)
HOP WITH $N = 200$ & $m = 1.1$	58.8 (4.4)	**64.3 (3.8)**	**68.2 (2.5)**	**69.5 (2.3)**	**71.2 (2.4)**
POISSON	**60.8 (4.6)**	**66.1 (3.9)**	**69.6 (2.6)**	**71.2 (2.2)**	**72.4 (2.3)**

Table 5.6: Cifar-10: Mean (standard deviation) classification accuracy over 100 trials with $N = 200$.

# LABELS PER CLASS	**1**	**2**	**3**	**4**	**5**
HOP WITH $N = 200$ & $m = 1.6$	**39.8 (5.3)**	45.0 (5.0)	48.2 (3.6)	50.3 (3.4)	51.8 (3.0)
HOP WITH $N = 200$ & $m = 1.3$	39.3 (5.1)	**45.5 (4.9)**	49.2 (3.4)	51.7 (3.1)	53.4 (2.7)
HOP WITH $N = 200$ & $m = 1.2$	38.9 (5.0)	45.3 (4.8)	**49.2 (3.4)**	**51.8 (3.1)**	**53.6 (2.7)**
POISSON	**40.7 (5.5)**	**46.5 (5.1)**	**49.9 (3.4)**	**52.3 (3.1)**	**53.8 (2.6)**

Therefore, as we observe from Table 5.4, 5.5 and 5.6, the MinMax of the algorithm classification accuracy happens when $m \geq 1$ for N large. In particular, this means that the performance of the higher order Poisson learning is optimized with $m \geq 1$ when the solution of (4.2) is more accurately approximated. Moreover, we remark that the standard Poisson learning outperforms HOP in every case when N is large. This also corroborates our analysis earlier about the positive impact on the accuracy scores brought by lower spectral cutoffs N.

Comparing the performance of the HOP algorithm with other semi-supervised machine learning algorithm evaluated in [4], we note that the higher order Poisson learning performs relatively better

than the majority of semi-supervised learning algorithms under very low label rates. In particular, we observe that the accuracy score of the higher order Poisson learning algorithm is comparable and at times higher than that of the standard Poisson learning. From the discussion in the earlier parts of this section, we conclude that our higher order Poisson learning algorithm outperforms Laplacian learning, Nearest Neighbor, Random Walk, MBO, WNLL, Centered Kernal, Sparse LP and p-Laplacian learning under very low label rates. Moreover, the accuracy score of the higher order Poisson learning is equivalent to that of Poisson learning for most of the times, while there are other times the higher order Poisson learning slightly out-performs Poisson learning due to the hyper-parameter from the spectral cutoff N. In general, the higher order Poisson algorithm provides the advantage of flexibility due to the unlimited choice for its parameters N and m. However, it potentially suffers from overfitting based on the choice of m, and its computational complexity is intractable without using some forms of approximation, such as spectral approximation.

Conclusion

We proved the existence and uniqueness of solutions for the higher order Poisson learning problems. We then developed the higher order Poisson (HOP) learning algorithm using spectral cutoff to avoid intractable computational complexity. We observed that the HOP learning algorithm with Poisson leaning loss performs similarly to Poisson learning developed in [4] and outperforms most of the algorithms evaluated in [4]. Moreover, the HOP learning algorithm provides the advantage of better flexibility due to its extra hyper-parameter called the spectral cutoff N from the spectral approximation. In order to more efficiently implement the HOP learning algorithm for the best result, a reasonable indicator for choosing the parameter values of N and m should be investigated further in future studies.

Appendix

We present here the complete visualization of the simulation results.

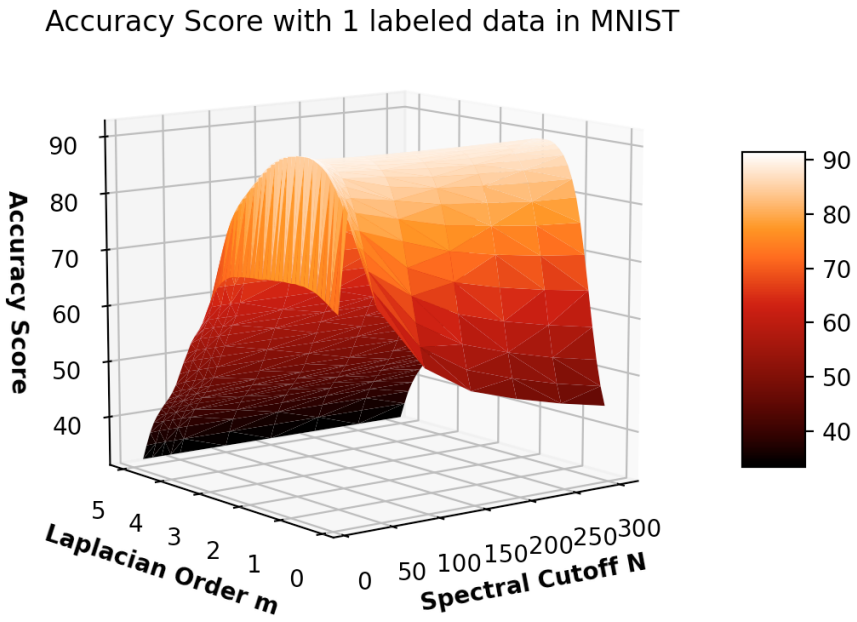

Figure A.1: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in MNIST.

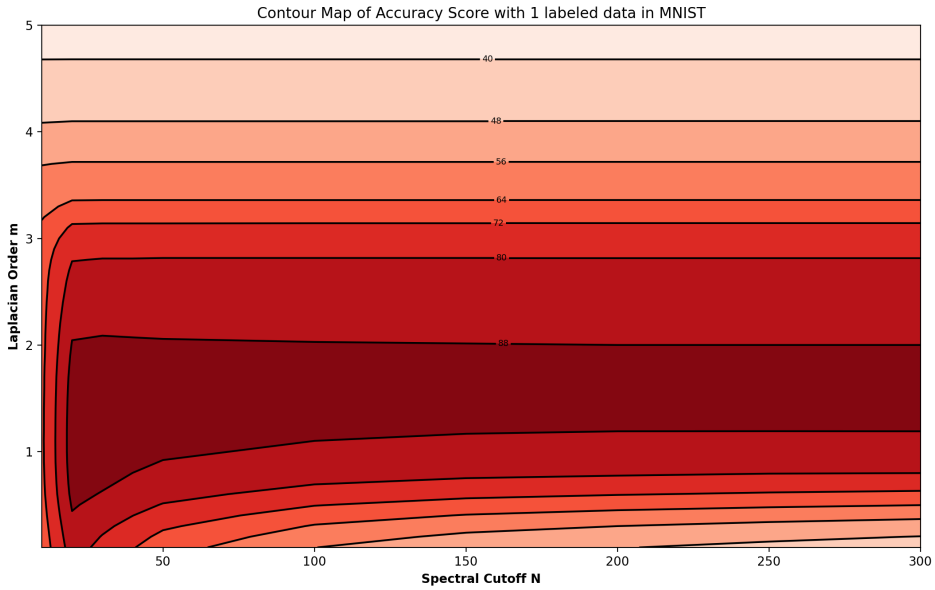

Figure A.2: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in MNIST.

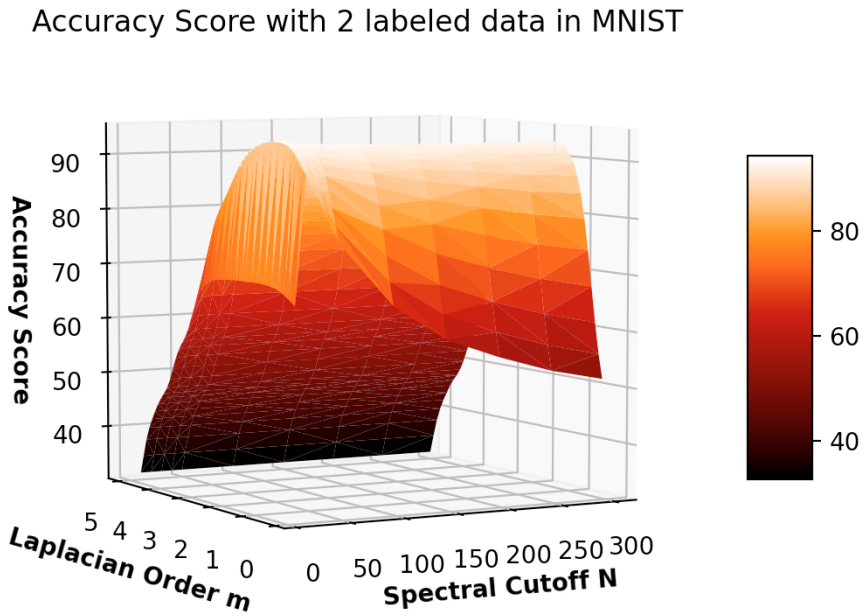

Figure A.3: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for two labeled data per class in MNIST.

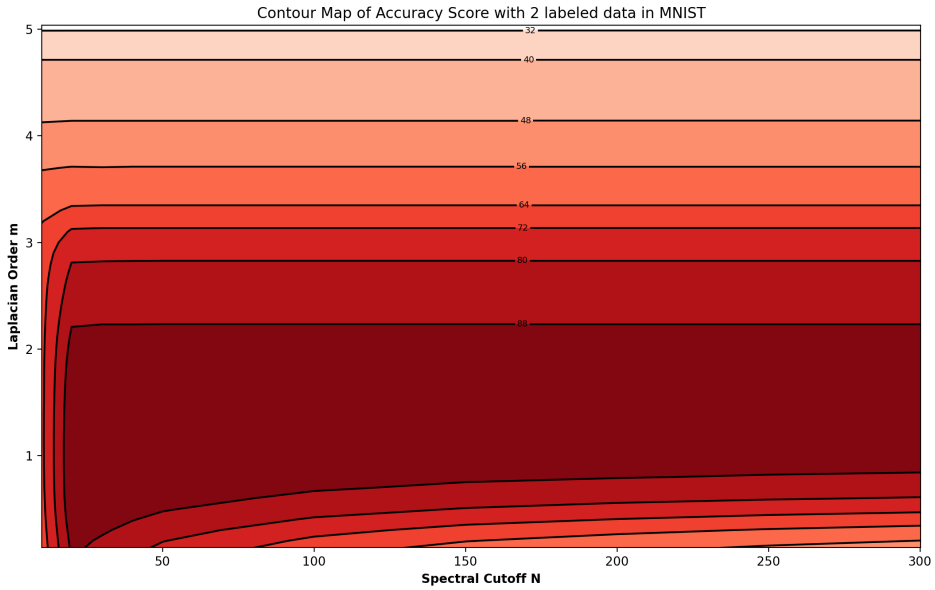

Figure A.4: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for two labeled data per class in MNIST.

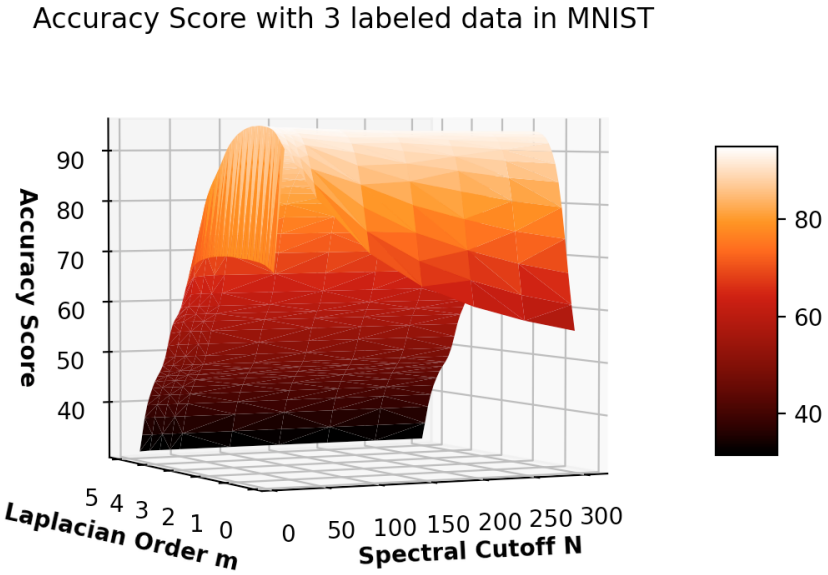

Figure A.5: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for three labeled data per class in MNIST.

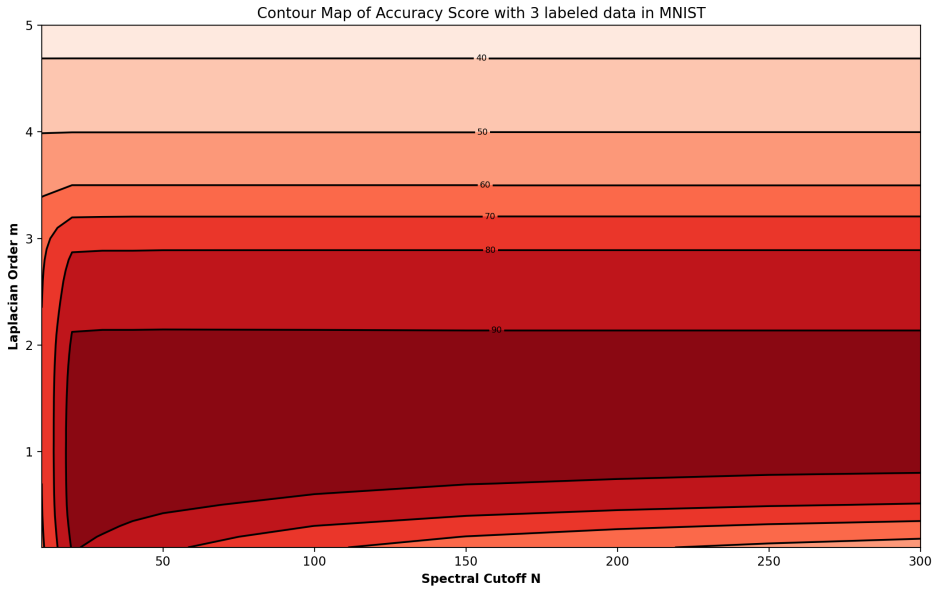

Figure A.6: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for three labeled data per class in MNIST.

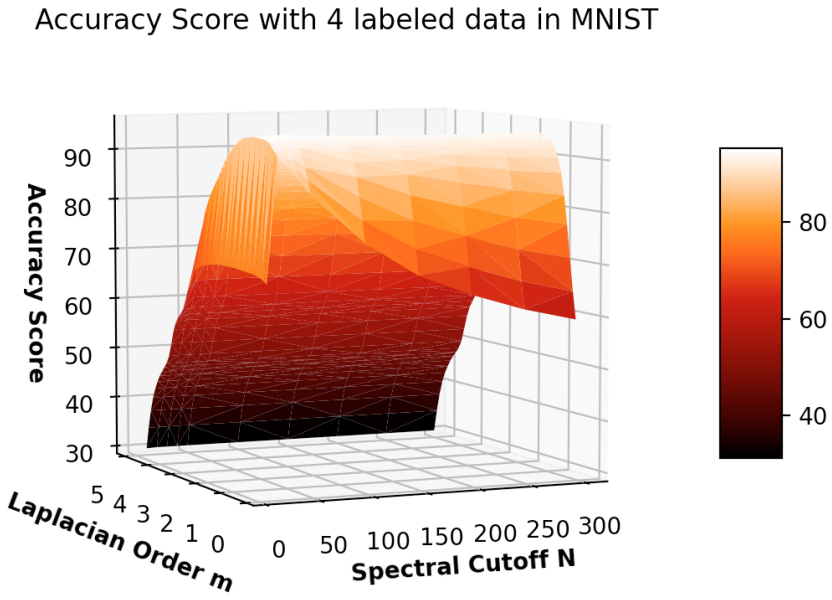

Figure A.7: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for four labeled data per class in MNIST.

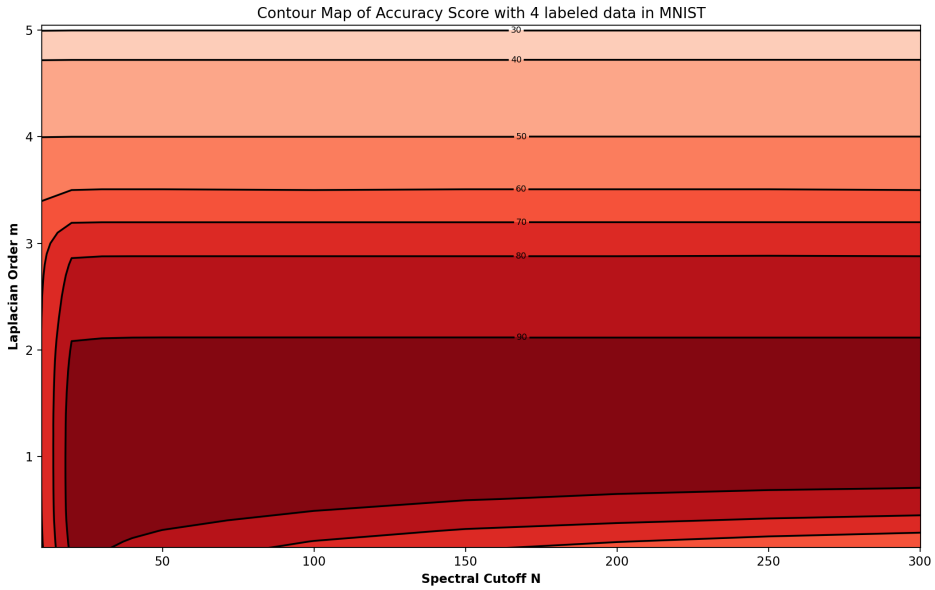

Figure A.8: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for four labeled data per class in MNIST.

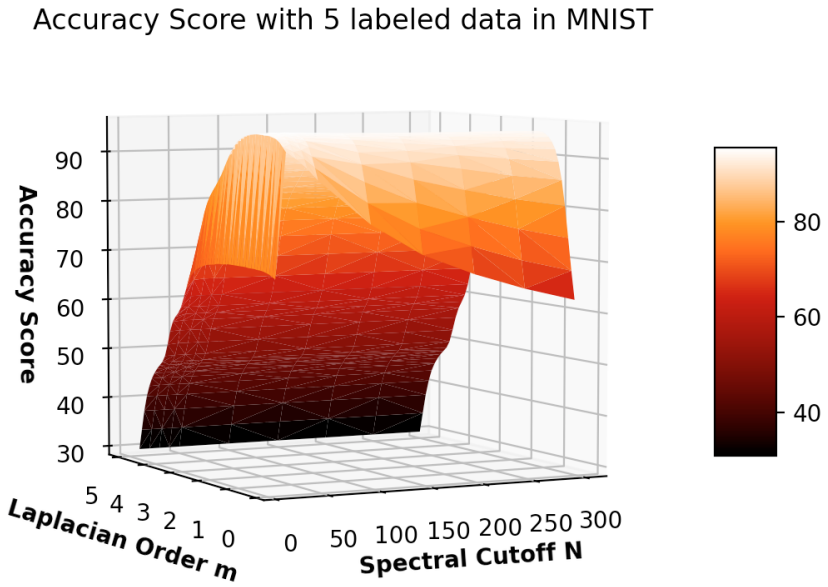

Figure A.9: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for five labeled data per class in MNIST.

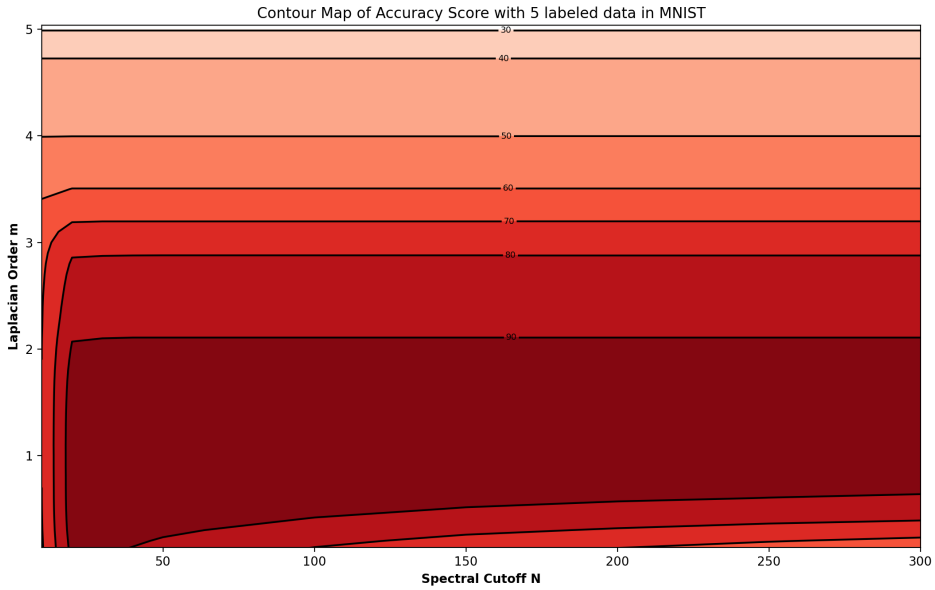

Figure A.10: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for five labeled data per class in MNIST.

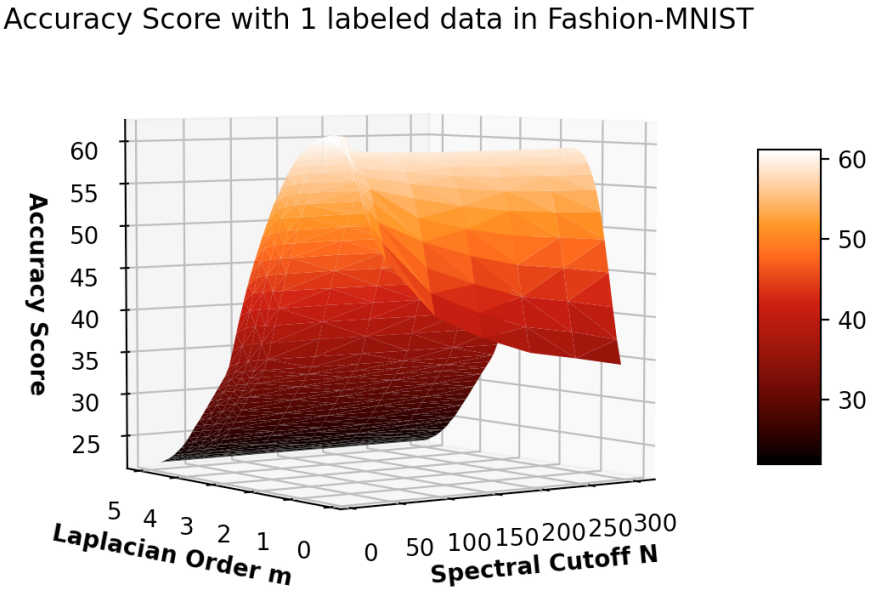

Figure A.11: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in Fashion-MNIST.

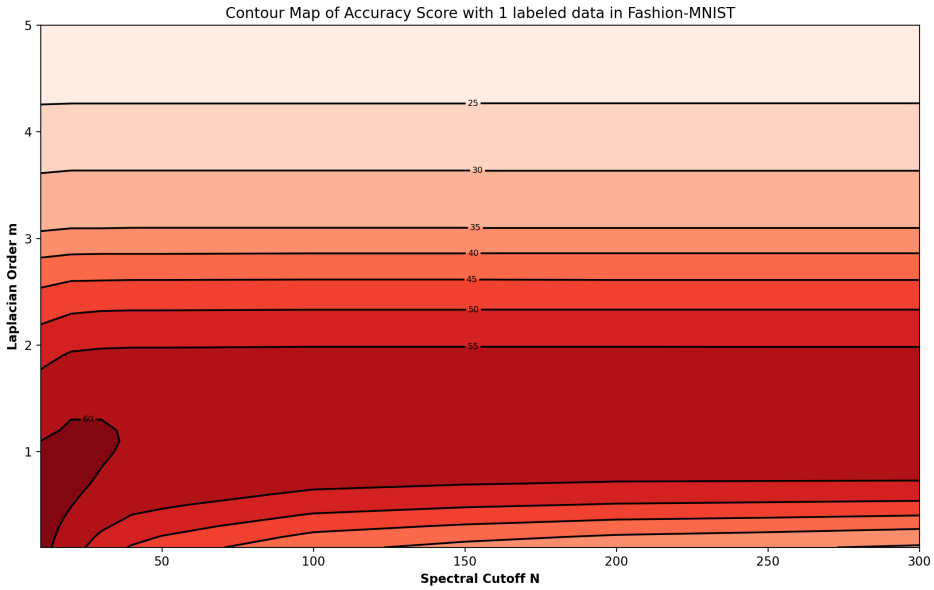

Figure A.12: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in Fashion-MNIST.

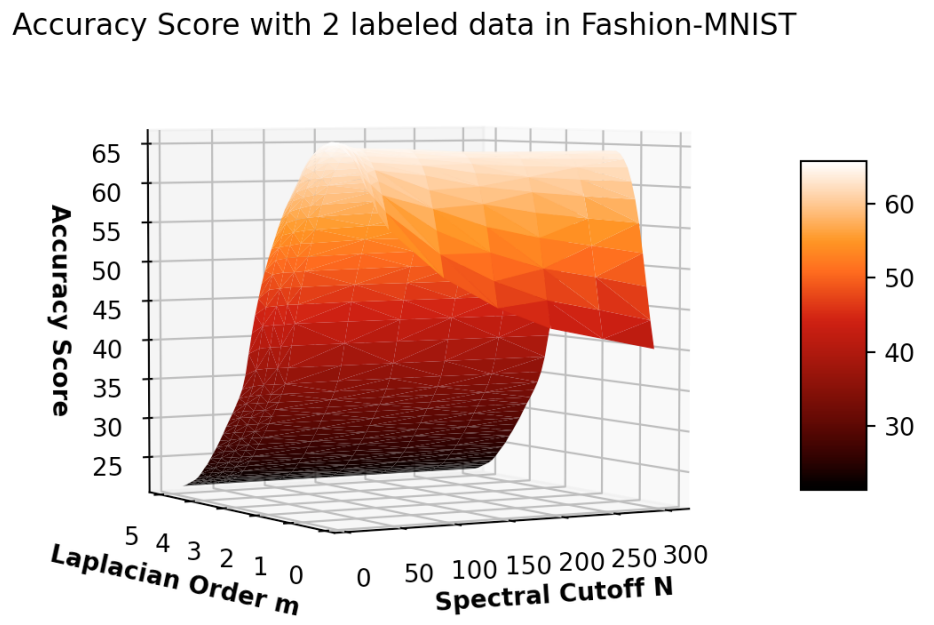

Figure A.13: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for two labeled data per class in Fashion-MNIST.

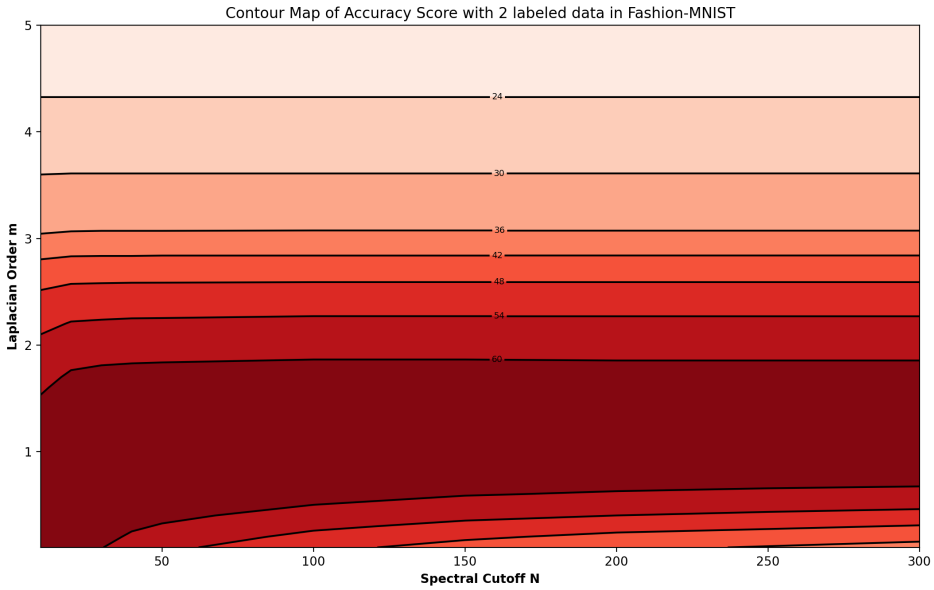

Figure A.14: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for two labeled data per class in Fashion-MNIST.

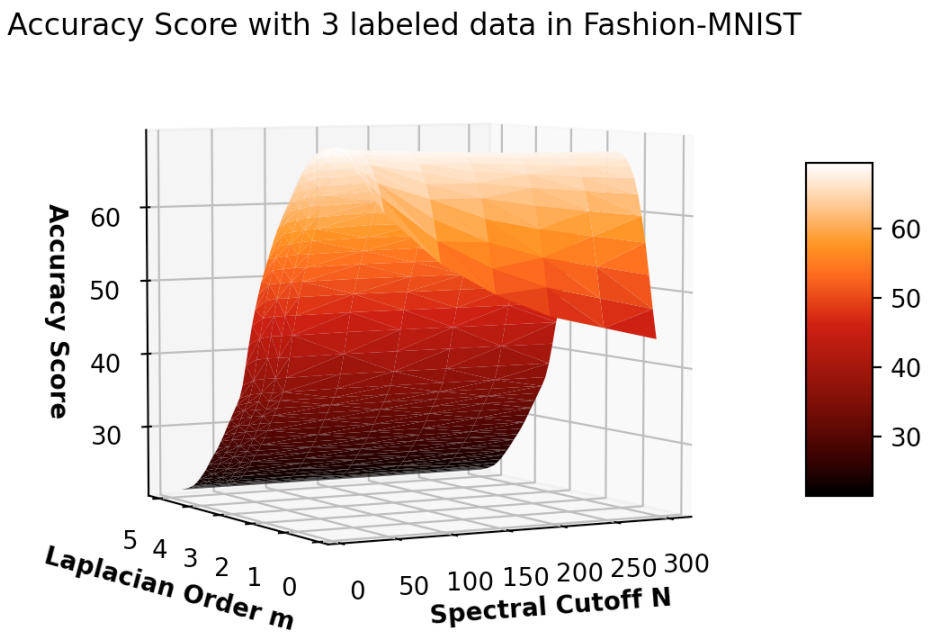

Figure A.15: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for three labeled data per class in Fashion-MNIST.

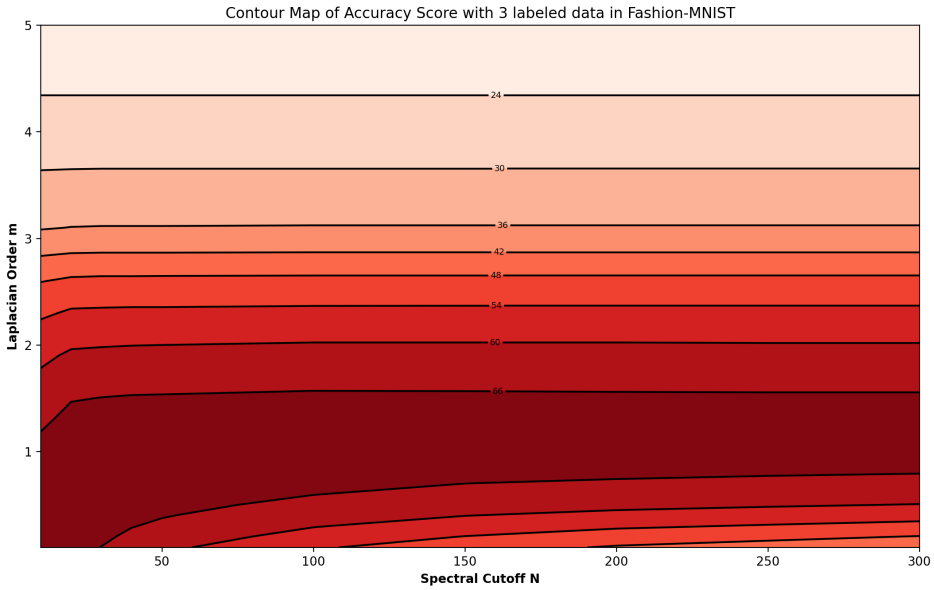

Figure A.16: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for three labeled data per class in Fashion-MNIST.

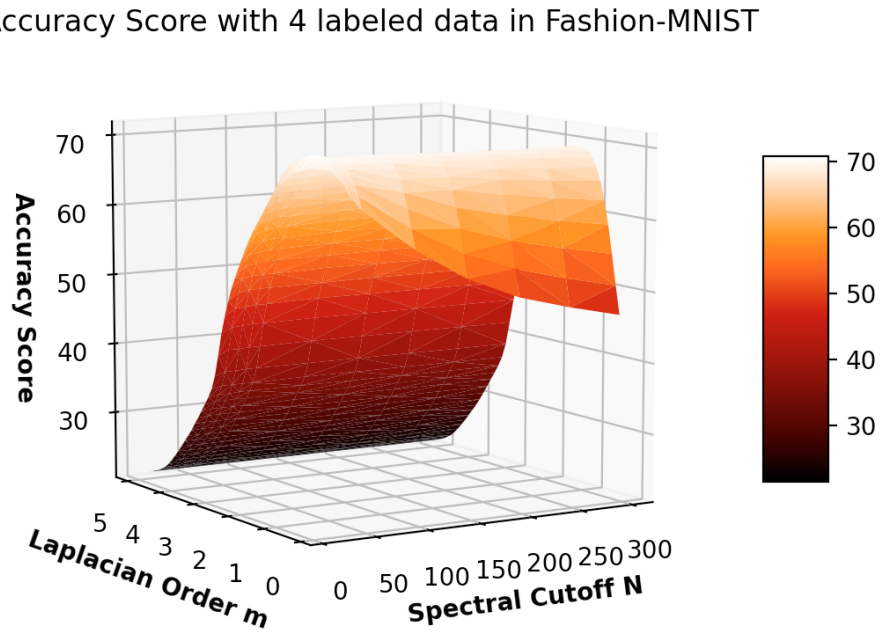

Figure A.17: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for four labeled data per class in Fashion-MNIST.

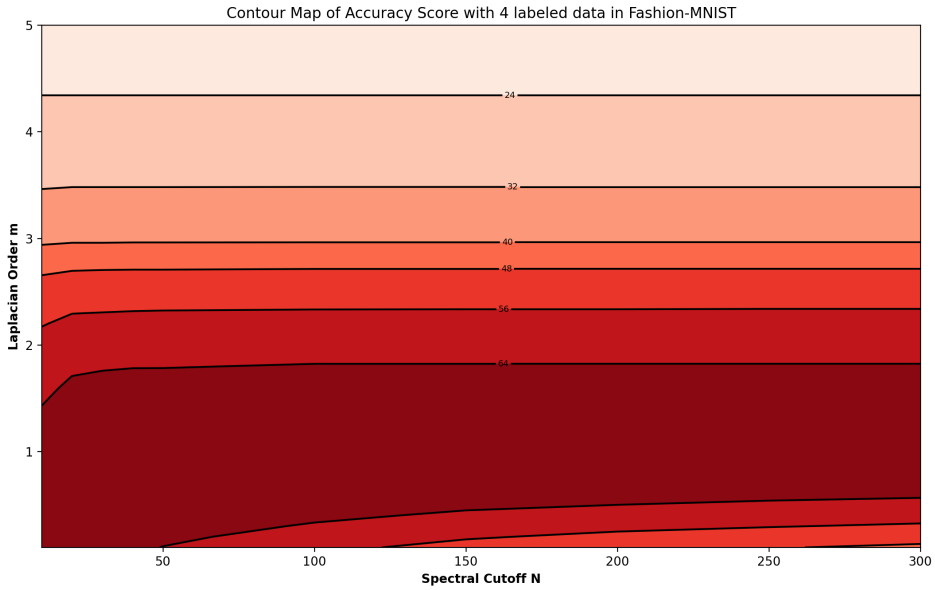

Figure A.18: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for four labeled data per class in Fashion-MNIST.

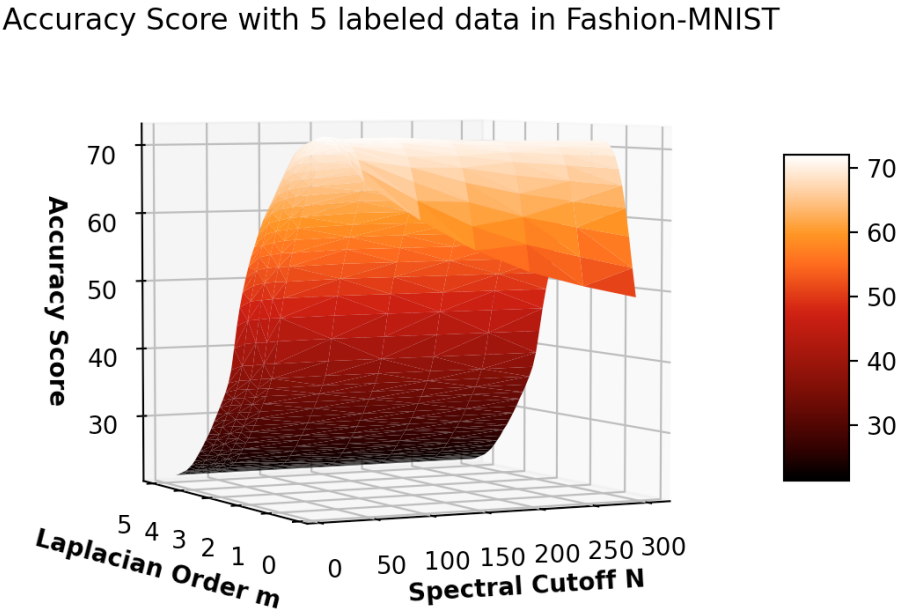

Figure A.19: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for five labeled data per class in Fashion-MNIST.

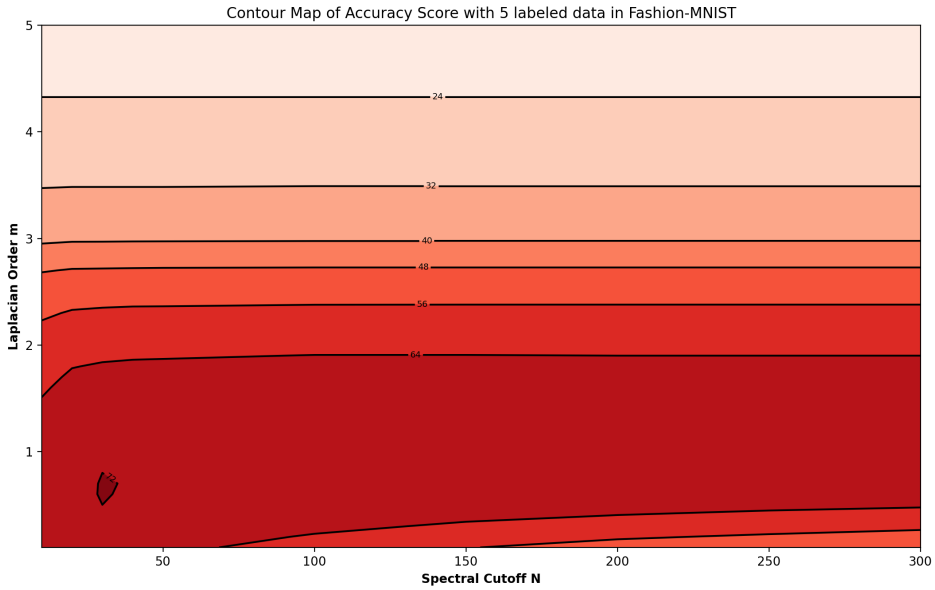

Figure A.20: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for five labeled data per class in Fashion-MNIST.

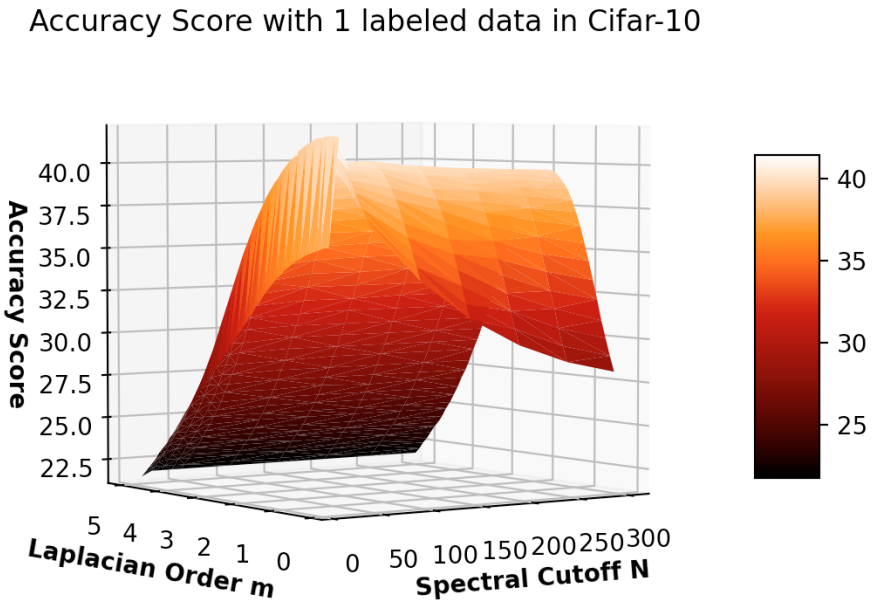

Figure A.21: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in Cifar-10.

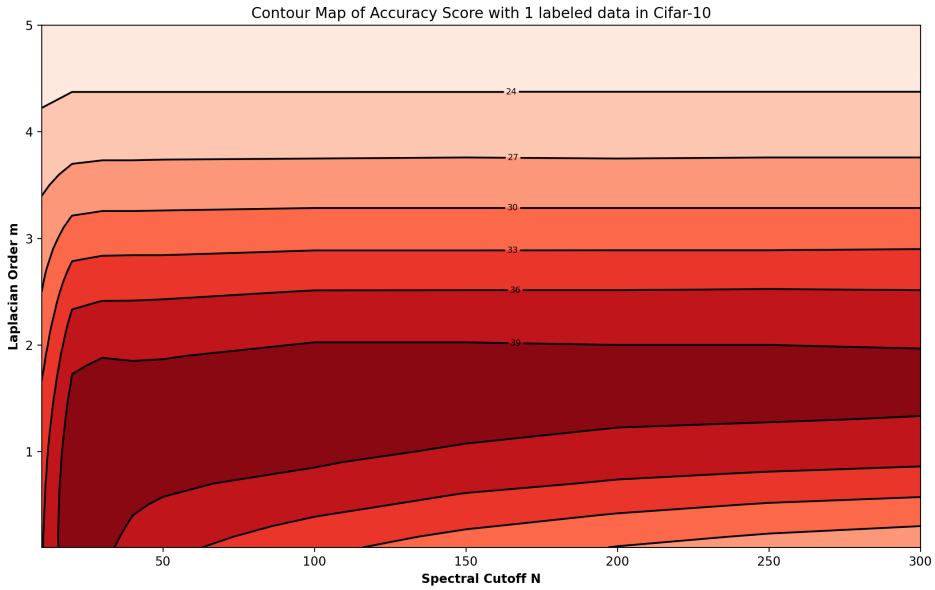

Figure A.22: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for a single labeled data per class in Cifar-10.

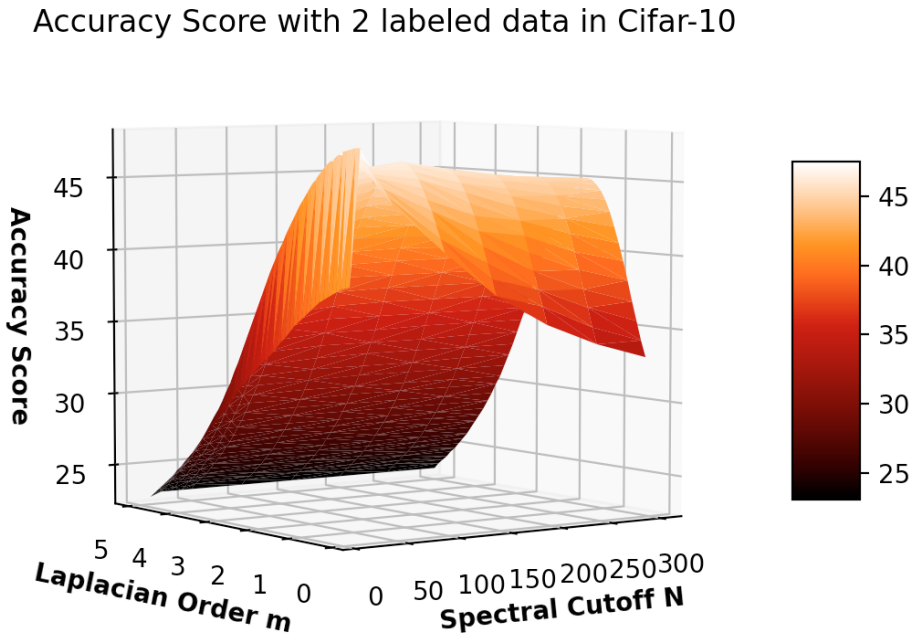

Figure A.23: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for two labeled data per class in Cifar-10.

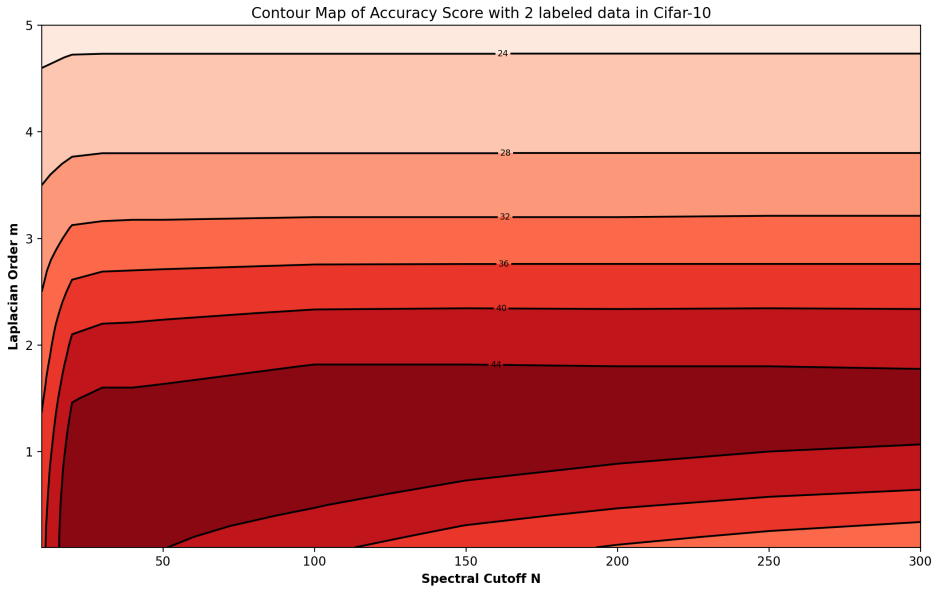

Figure A.24: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for two labeled data per class in Cifar-10.

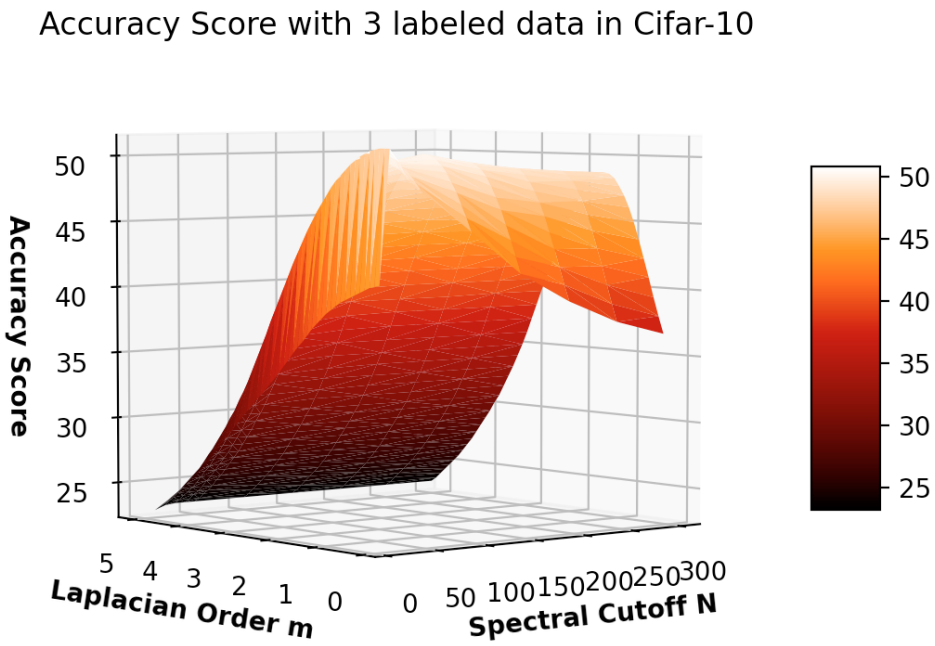

Figure A.25: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for three labeled data per class in Cifar-10.

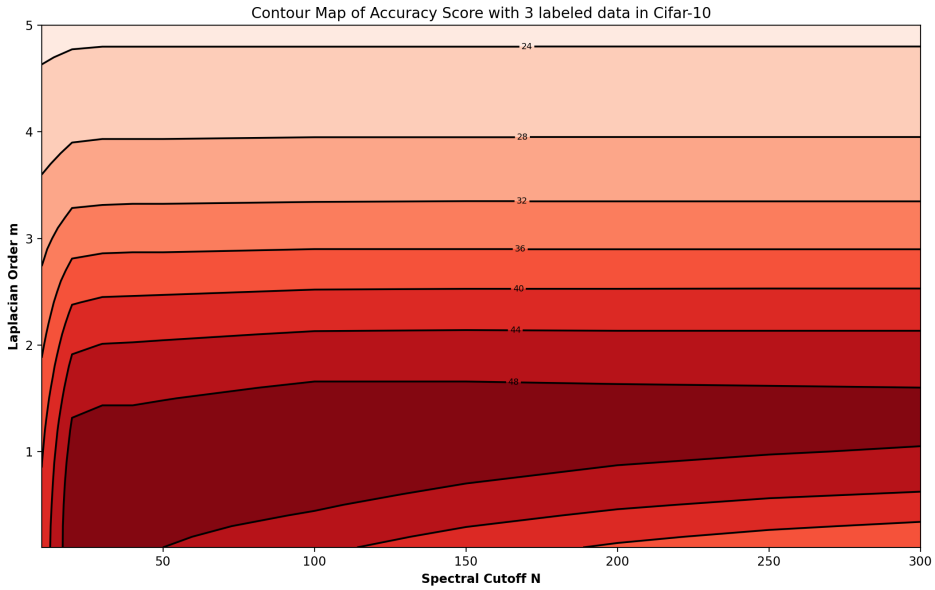

Figure A.26: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for three labeled data per class in Cifar-10.

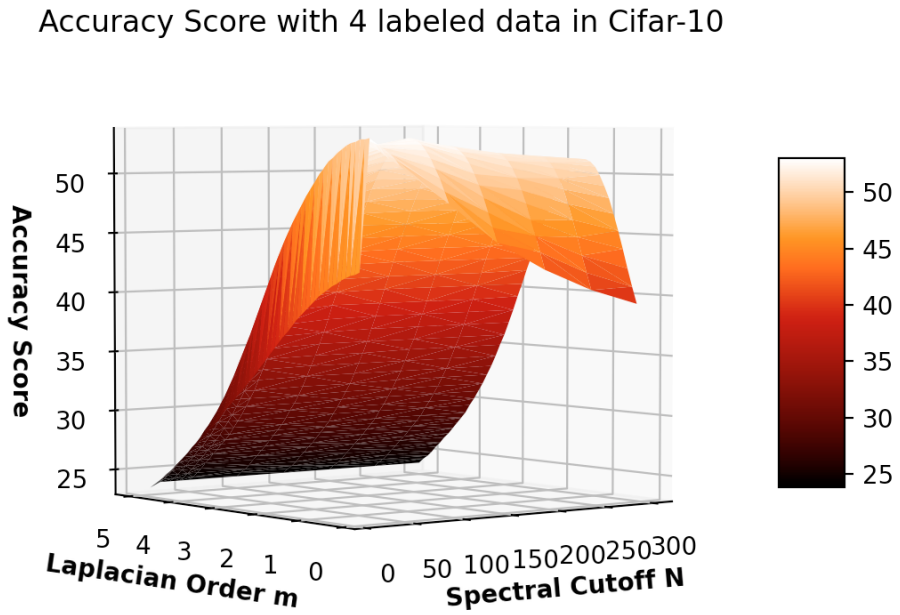

Figure A.27: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for four labeled data per class in Cifar-10.

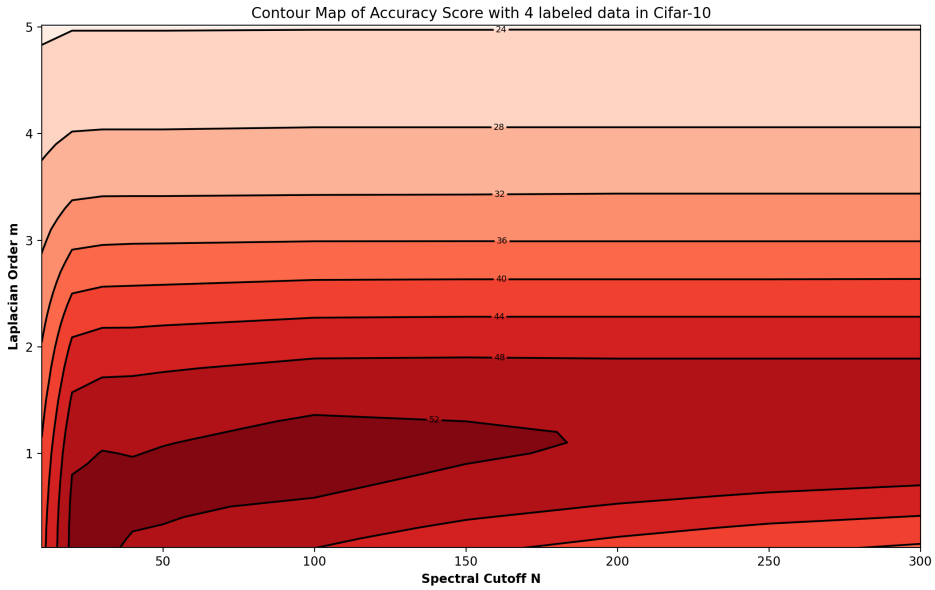

Figure A.28: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for four labeled data per class in Cifar-10.

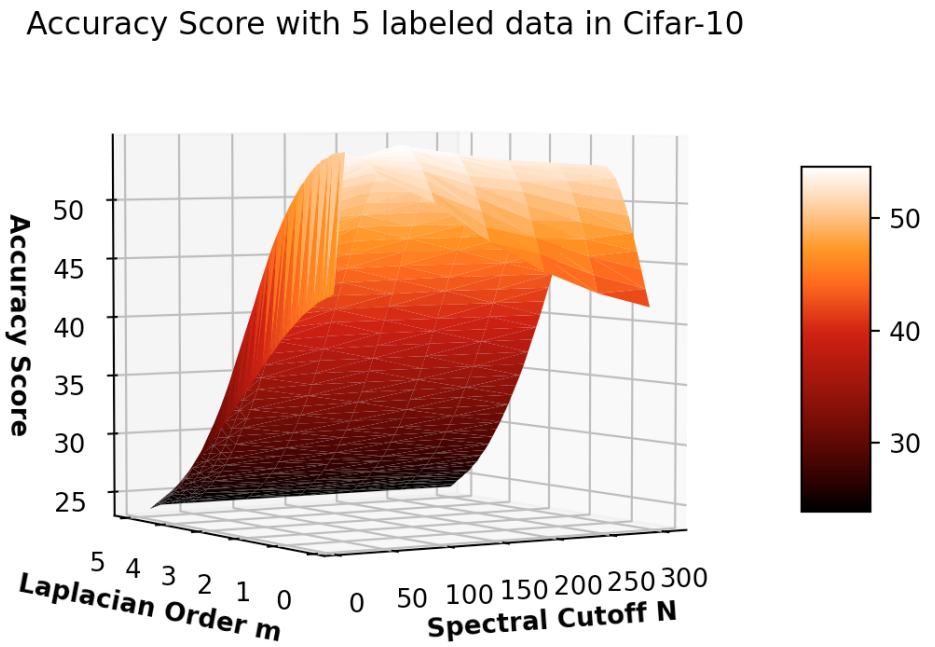

Figure A.29: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for five labeled data per class in Cifar-10.

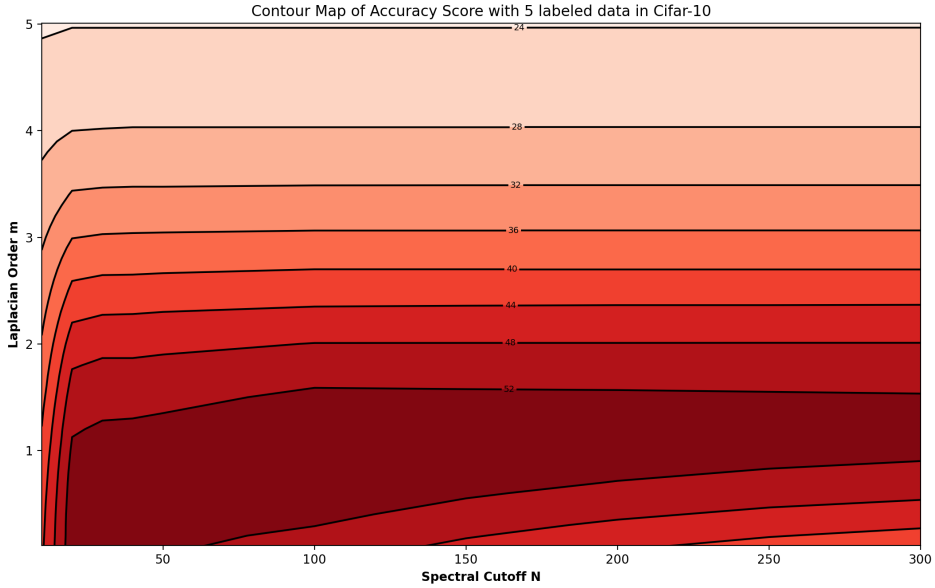

Figure A.30: The mean accuracy score of the higher order Poisson learning with combination of order m and spectral cutoffs N for five labeled data per class in Cifar-10.

References

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

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

[3] J. Calder. The calculus of variations, 2020. Online Lecture Notes: http://www-users.math.umn.edu/~jwcalder/CalculusOfVariations.pdf.

[4] J. Calder, B. Cook, M. Thorpe, and D. Slepčev. Poisson Learning: Graph Based semi-supervised learning at very low label rates. *Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119*, 2020.

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

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

[7] M. M. Dunlop, D. Slepčev, A. M. Stuart, and M. Thorpe. Large data and zero noise limits of graph-based semi-supervised learning algorithms. *Applied and Computational Harmonic Analysis*, 2019.

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

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