# Efficiently Learning the Graph for Semi-supervised Learning

**Dravyansh Sharma**[1] **Maxwell Jones**[1]

[1]School of Computer Science., Carnegie Mellon University, Pittsburgh, PA, 15213

## Abstract

Computational efficiency is a major bottleneck in using classic graph-based approaches for semi-supervised learning on datasets with a large number of unlabeled examples. Known techniques to improve efficiency typically involve an approximation of the graph regularization objective, but suffer two major drawbacks – first the graph is assumed to be known or constructed with heuristic hyperparameter values, second they do not provide a principled approximation guarantee for learning over the full unlabeled dataset. Building on recent work on learning graphs for semi-supervised learning from multiple datasets for problems from the same domain, and leveraging techniques for fast approximations for solving linear systems in the graph Laplacian matrix, we propose algorithms that overcome both the above limitations.

We show a formal separation in the learning-theoretic complexity of sparse and dense graph families. We further show how to approximately learn the best graphs from the sparse families efficiently using the conjugate gradient method. Our approach can also be used to learn the graph efficiently online with sub-linear regret, under mild smoothness assumptions. Our online learning results are stated generally, and may be useful for approximate and efficient parameter tuning in other problems. We implement our approach and demonstrate significant ($\sim$10-100x) speedups over prior work on semi-supervised learning with learned graphs on benchmark datasets.

## 1 INTRODUCTION

As machine learning finds applications in new domains like healthcare, finance and a variety of industrial sectors [Vamathevan et al., 2019, Kumar et al., 2022, Larrañaga et al., 2018], obtaining sufficiently large human-annotated datasets for applying supervised learning is often prohibitively expensive. Semi-supervised learning can solve this problem by utilizing unlabeled data, which is more readily available, together with a small amount of human-labeled data. Graph-based techniques, where the similarity of examples is encoded using a graph, are popular and effective for learning using unlabeled data [Zhu and Goldberg, 2009]. Several heuristic approaches for learning given the graph are known, but the choice of a good graph is strongly dependent on the problem domain. How to create the graph has largely been 'more of an art than science' [Zhu, 2005], although recent work proposes how to provably learn the best graph for a given problem domain from the data [Balcan and Sharma, 2021]. A key limitation of the proposed techniques is their computational efficiency, as the proposed algorithms take $\tilde{O}(n^4)$ time which make them impractical to run on real datasets. In this work we propose new and more practical approaches that exploit graph *sparsity* and employ *approximate optimization* to obtain more powerful graph learning techniques with formal guarantees for their effectiveness, and improved efficiency guarantees.

Past work on improving the efficiency of graph-based semi-supervised learning has focused largely on selecting a subset of 'important' unlabeled examples. For example, one may use a greedy algorithm [Delalleau et al., 2005] or a $k$-means based heuristic [Wang et al., 2016], run the graph-based algorithm only on the selected subset of examples and use some local interpolation for remaining nodes. In this work we provide more principled approaches that come with formal near-optimality guarantees, and demonstrate the trade-off between accuracy and efficiency. We employ the conjugate gradient method to compute fast, approximate inverses and optimize over new multi-parameter graph families that include sparse graphs which can be more efficiently optimized. Our work extends prior theoretical results [Balcan and Sharma, 2021] on sample efficiency to additional graph families that capture sparsity. Empirically, we observe

*Accepted for the 39th Conference on Uncertainty in Artificial Intelligence* (UAI 2023).

that our proposed approaches are computationally efficient, while retaining the effectiveness guarantees of learning the best graph for the given problem distribution.

More precisely, we exhaustively learn the graph hyperparameter for the commonly used Gaussian kernel over a continuous interval. This approach is stronger than a grid search, which only computes results at some finite set of hyperparameter values. We extend the recent line of work on data-driven algorithm design (Section 1.2) to approximate online feedback, and achieve provably near-optimal hyperparameter selection.

### 1.1 MAIN CONTRIBUTIONS

- (Section 3) We provide a general analysis for online data-driven algorithm design with approximate loss functions, quantifying the accuracy-efficiency trade-off. While prior work on approximate algorithm selection [Balcan et al., 2020e] studies bounded $\ell_\infty$-norm approximations in the distributional setting, we generalize along two axes – we study a more general approximation class necessary to analyse our semi-supervised learning algorithm, and our results apply to online learning, even in the presence of (the more realistic) partial feedback.
- (Section 4) For graph-based semi-supervised learning, we show a formal gap in the pseudodimension of learning sparse and dense graphs. Concretely, if each graph node is connected to at most $K$ neighbors, the pseudodimension is $O(K+\log n)$, which implies an asymptotic gap relative to $\Omega(n)$ bound for learning complete graphs with Gaussian RBF kernels [Balcan and Sharma, 2021].
- (Section 5) We propose an efficient algorithm based on approximate Laplacian inverse for approximately computing the hyperparameter intervals where the semi-supervised loss objective is constant. We prove convergence guarantees for our algorithm, which capture a trade-off between the computational efficiency and the accuracy of loss estimation. We instantiate our approach for approximate graph learning for the classic harmonic-objective algorithm of Zhu et al. [2003], as well as the computationally efficient algorithm of Delalleau et al. [2005].
- (Section 6) We implement our algorithm [1] and provide extensive empirical study showing improvement over previously proposed approaches on standard datasets. Specifically, we improve the running time by about 1-2 orders of magnitude, while almost retaining (and in some cases slightly increasing) the accuracy.

### 1.2 RELATED WORK

**Approximate Laplacian inverse**. The *conjugate gradient method* [Hestenes and Stiefel, 1952] is an iterative algorithm used to approximately solve a system $Ax = b$ for symmetric, positive definite matrices. Starting with the zero vector as an approximate solution, every iteration computes a gradient used to update this approximation in the direction of the exact solution. The exact solution itself is obtained in $n$ steps, but good approximate solutions can be found much sooner for graphs with low condition number $\kappa$ [Axelsson, 1976, Vishnoi, 2012]. Furthermore, each iteration computes a finite number of matrix-vector products on $A$, yielding good runtime guarantees. Many variants of the Conjugate gradient method exist [Hager and Zhang, 2006], in this work we use the original version. The conjugate gradient method is a tool in use for calculating fast matrix inverses across machine learning applications, in domains such as deep reinforcement learning [Schulman et al., 2015, Rajeswaran et al., 2017] and market forecasting [Shen et al., 2015]. We choose the conjugate gradient method over other iterative techniques to solve $Ax = b$ like Lanczos iteration due to its stability, simplicity, and previous success in other machine learning applications. Further, the conjugate gradient method offers strong theoretical guarantees, leading to fast approximate convergence for our use case.

**Semi-supervised learning**. *Semi-supervised learning* is a paradigm for learning from labeled and unlabeled data ([Zhu and Goldberg, 2009, Balcan and Blum, 2010]). A popular approach for semi-supervised learning is to optimize a graph-based objective. Several methods have been proposed to predict labels *given a graph* including $st$-mincuts ([Blum and Chawla, 2001]), soft mincuts that optimize a harmonic objective ([Zhu et al., 2003]), and label propagation ([Zhu and Ghahramani, 2002]). Prior research for efficient semi-supervised learning has also typically assumed that the graph $G$ is given [Delalleau et al., 2005, Wang et al., 2016]. All algorithms have comparable performance provided the graph $G$ encodes the problem well [Zhu and Goldberg, 2009]. Balcan and Sharma [2021] introduce a first approach to learn the graph $G$ with formal guarantees, and show that the performance of all the algorithms depends strongly on the graph hyperparameters. In this work, we provide computationally efficient algorithms for learning the graph parameters. While we focus on learning the graph more classic approaches, deep learning based approaches also typically assume a graph is available Kipf and Welling [2017].

**Data-driven algorithm design**. Gupta and Roughgarden [2017] define a formal learning framework for selecting algorithms from a family of heuristics or setting hyperparameters. It is further developed by Balcan et al. [2017] and surveyed in Balcan [2020]. It has been successfully applied to several combinatorial problems like integer programming and clustering [Balcan et al., 2018a,c] and for giving powerful guarantees like adversarial robustness, adaptive learning and differential privacy [Balcan et al., 2020a, 2018b, 2020c,d]. Balcan et al. [2018b, 2020b] introduce

[1] https://github.com/maxwelljones14/Efficient-SSL

general data-driven design techniques under some smoothness assumptions, and Balcan et al. [2020e] study learning with approximate losses. We extend the techniques to significantly broader problem settings (as noted in Section 1.1), and investigate the structure of graph-based label learning formulation to apply the new techniques.

## 2 NOTATION AND FORMAL SETUP

We are given some labeled points $L$ and unlabeled points $U$. One constructs a graph $G$ by placing (possibly weighted) edges $w(u, v)$ between pairs of data points $u, v$ which are 'similar', and labels for the unlabeled examples are obtained by optimizing some graph-based score. We have an oracle $O$ which on querying provides us the labeled and unlabeled examples, and we need to pick $G$ from some family $\mathcal{G}$ of graphs. We commit to using some algorithm $A(G, L, U)$ (or $A_{G,L,U}$) which provides labels for examples in $U$, and we should pick a $G$ such that $A(G, L, U)$ results in small error in its predictions on $U$. To summarize more formally,

*Problem statement*: Given data space $\mathcal{X}$, label space $\mathcal{Y}$ and an oracle $O$ which yields a number of labeled examples $\emptyset \neq L \subset \mathcal{X} \times \mathcal{Y}$ and some unlabeled examples $\emptyset \neq U \subset \mathcal{X}$ such that $|L| + |U| = n$. We are further given a parameterized family of graph construction procedures over parameter space $\mathcal{P}$, $\mathcal{G} : \mathcal{P} \to (\mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0})$, graph labeling algorithm $A_{G,L,U}$ which takes a graph $G$ with labeled nodes $L$ and unlabeled nodes $U$ and provides labels for all unlabeled examples in $U$, a loss function $l : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ and a target labeling $\tau : U \to \mathcal{Y}$. We need to select $\rho \in \mathcal{P}$ such that corresponding graph $G(\rho)$ minimizes $\frac{1}{|U|} \sum_U l(A_{G(\rho),L,U}(u), \tau(u))$ w.r.t. $\rho$.

We will now describe graph families $\mathcal{G}$ and algorithms $A_{G,L,U}$ considered in this work. We restrict our attention to binary classification, i.e. $\mathcal{Y} = \{0, 1\}$, and note that all proposed algorithms naturally extend to multiclass problems, using the standard one-vs-all trick. We assume there is a feature based *similarity function* $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$, a metric which monotonically captures similarity between the examples. In Definition 1 we formally introduce parametric families to build a graph using the similarity function, which capture and interpolate well-known approaches such as $k$-nearest neighbor graphs, $r$-neighborhood graphs and Gaussian RBF kernels. In this work, we will consider three parametric families of graph construction algorithms defined below. $\mathbb{I}[\cdot]$ is the indicator function taking values in $\{0, 1\}$. Let $N_k(v)$ denote the set of nodes of $G$ which are the $k$-nearest neighbors of node $v$ under the metric $d(\cdot, \cdot)$. Define $k$-mutual neighborhood as the set of edges for which each end-point is a $k$-nearest neighbor of the other, i.e. $N'_k = \{(u, v) \mid u \in N_k(v) \text{ and } v \in N_k(u)\}$ [Ozaki et al., 2011].

**Definition 1.** *Sparse graph families.*
*a) Thresholded nearest neighbors, $G(k, r)$, (with $k \in \mathbb{Z}^+, r \in \mathbb{R}^+$): $w(u, v) = \mathbb{I}[d(u, v) \leq r$ and $(u, v) \in N'_k]$.*
*b) Gaussian nearest neighbors, $G(k, \sigma)$, (with $k \in [K]$ for $K \in \mathbb{Z}^+$, $\sigma \in \mathbb{R}^+$): $w(u, v) = e^{-\frac{d(u,v)^2}{\sigma^2}} \mathbb{I}[(u, v) \in N'_k]$.*

The thresholded nearest neighbor graph adds unweighted edges to $G(k, r)$ only when the examples are closer than some $r \in \mathbb{R}_{\geq 0}$, and are mutual $k$-nearest neighbors. The Gaussian (or RBF) kernel is more powerful and allows weighted edges that depend on the metric distance and the bandwidth parameter $\sigma$. We will use $\rho$ to denote a general graph parameter (e.g. $\rho = (k, r)$ for thresholded nearest neighbors) and denote the general parameterized graph family by $G(\rho)$. Once the graph is constructed using one of the above families, we can assign labels using a suitable algorithm $A_{G,L,U}$. A popular and effective approach is by optimizing a quadratic objective $\frac{1}{2} \sum_{u,v} w(u, v)(f_u - f_v)^2 = f^T(D - W)f$. Here $f$ may either be discrete $f_v \in \{0, 1\}$ which corresponds to finding a graph mincut separating the oppositely labeled vertices Blum and Chawla [2001], or $f$ may be continuous, i.e. $f \in [0, 1]$, and we can round $f$ to obtain the labels [Zhu et al., 2003].

In the *distributional setting*, we are presented with several instances of the graph semi-supervised learning problem assumed to be drawn from an unknown distribution $\mathcal{D}$ and want to learn the best value of the graph parameter $\rho$. We also assume we get all the labels for the 'training' problem instances. A choice of $\rho$ uniquely determines the graph $G(\rho)$ and we use some algorithm $A_{G(\rho),L,U}$ to make predictions (e.g. minimizing the quadratic penalty score above) and suffers loss $l_{A(G(\rho),L,U)} := \frac{1}{|U|} \sum_U l(A_{G(\rho),L,U}(u), \tau(u))$ which we seek to minimize relative to smallest possible loss by some graph in the hypothesis space, in expectation over the data distribution $\mathcal{D}$. We also define the family of loss functions $\mathcal{H}_\rho = \{l_{A(G(\rho),L,U)} \mid \rho \in \mathcal{P}\}$. For example, $\mathcal{H}_{k,r} = \{l_{A(G(k,r),L,U)} \mid (k, r) \in \mathbb{Z}^+ \times \mathbb{R}^+\}$. As a shorthand, we will often denote the loss on a fixed problem instance as a function of the graph hyperparameter $\rho$ as simply $l(\rho)$, and refer to it as the *dual semi-supervised loss*.

Finally we note definitions of some useful learning theoretic complexity measures. First recall the definition of pseudodimension [Pollard, 2012] which generalizes VC dimension to real-valued functions, and is a well-known measure for hypothesis-space complexity in statistical learning theory.

**Definition 2** (Pseudo-dimension)**.** *Let $\mathcal{H}$ be a set of real valued functions from input space $\mathcal{X}$. We say that $C = (x_1, \ldots, x_m) \in \mathcal{X}^m$ is pseudo-shattered by $\mathcal{H}$ if there exists a vector $r = (r_1, \ldots, r_m) \in \mathbb{R}^m$ (called "witness") such that for all $b = (b_1, \ldots, b_m) \in \{\pm 1\}^m$ there exists $h_b \in \mathcal{H}$ such that sign$(h_b(x_i) - r_i) = b_i$. Pseudo-dimension of $\mathcal{H}$ is the cardinality of the largest set pseudo-shattered by $\mathcal{H}$.*

We will also need the definition of *dispersion* [Balcan et al., 2018b] which, informally speaking, captures how amenable

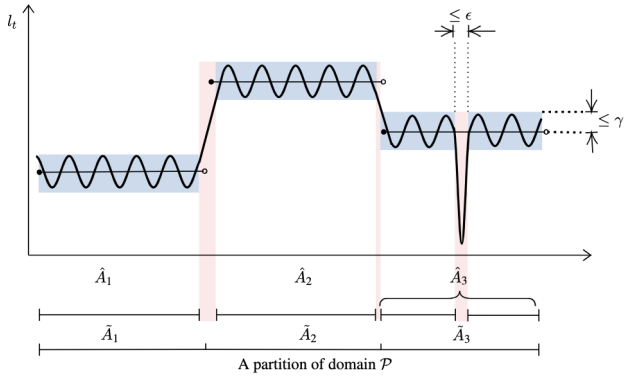



Figure 1: A depiction of $(\epsilon, \gamma)$-approximate feedback (Definition 4) for a one dimensional loss function. Here, the true loss $l_t$ is given by the solid curve, and approximate loss $\tilde{l}_t$ is piecewise constant.

a non-Lipschitz function is to online learning. As noted in Balcan et al. [2018b, 2020c], dispersion is necessary and sufficient for learning piecewise Lipschitz functions online.

**Definition 3** (Dispersion)**.** *The sequence of random loss functions $l_1, \ldots, l_T$ is $\beta$-dispersed for the Lipschitz constant $L$ if, for all $T$ and for all $\epsilon \geq T^{-\beta}$, we have that, in expectation, at most $\tilde{O}(\epsilon T)$ functions (here $\tilde{O}$ suppresses dependence on quantities beside $\epsilon, T$ and $\beta$, as well as logarithmic terms) are not $L$-Lipschitz for any pair of points at distance $\epsilon$ in the domain $\mathcal{C}$. That is, for all $T$ and $\epsilon \geq T^{-\beta}$,* $\mathbb{E}\left[\max_{\substack{\rho,\rho' \in \mathcal{C} \\ ||\rho-\rho'||_2 \leq \epsilon}} \left|\{t \in [T] \mid l_t(\rho) - l_t(\rho') > L||\rho-\rho'||_2\}\right|\right] \leq \tilde{O}(\epsilon T)$.

## 3 APPROXIMATE DATA-DRIVEN ALGORITHM DESIGN

Balcan and Sharma [2021] show that the dual loss $l(\rho)$ is a piecewise constant function of the graph hyperparameter $\rho$, for any fixed problem instance. Suppose the problem instances arrive *online* in rounds $t = 1, \ldots, T$, and the learner receives some feedback about her predicted parameter $\rho_t$. Since the *full information* setting where all the labels are revealed is not very practical (it assumes all labels of previous problem instances are available), Balcan and Sharma [2021] also consider the more realistic semi-bandit feedback setting of Balcan et al. [2020b], where only the loss corresponding to hyperparameter $\sigma_t$ selected by the online learner in round $t$ is revealed, along with the end points of the piece $A_t$ containing $\sigma_t$ where $l_t$ is contant. We consider a generalization of this setting where only an approximation to the loss value at $\sigma_t$ is revealed, along with an approximation to the piece $A_t$.

Although our formulation below is motivated by considerations for graph parameter tuning for semi-supervised learning, we provide very general definitions and results that apply to approximate online data-driven parameter selection more generally [Balcan, 2020].

**Definition 4.** *An online optimization problem with loss functions $l_1, l_2, \ldots$ is said to have $(\epsilon, \gamma)$-**approximate semi-bandit feedback** with system size $M$ if for each time $t = 1, 2, \ldots$, there is a partition $\tilde{A}_t^{(1)}, \ldots, \tilde{A}_t^{(M)}$ of the parameter space $\mathcal{P} \subset \mathbb{R}^d$, called an approximate feedback system, such that if the learner plays point $\rho_t \in \tilde{A}_t^{(i)}$, she observes the approximate feedback set $\tilde{A}_t^{(i)}$, and observes approximate loss $\tilde{l}_t(\rho)$ for all $\rho \in \tilde{A}_t^{(i)}$ such that $\sup_{\rho \in \hat{A}_t^{(i)}} |\tilde{l}_t(\rho) - l_t(\rho)| \leq \gamma$, for some (unknown) $\hat{A}_t^{(i)} \subseteq \tilde{A}_t^{(i)}$ with $\left|\mathsf{vol}\left(\tilde{A}_t^{(i)} \setminus \hat{A}_t^{(i)}\right)\right| \leq \epsilon$. Here $\mathsf{vol}(A)$ denotes the $d$-dimensional volume of set $A$. We let $\tilde{A}_t(\rho)$ denote the approximate feedback set that contains $\rho \in \mathcal{P}$.*

For example, let the parameter space $\mathcal{P}$ be one-dimensional, and in round $t$ the learner plays point $\rho_t \in \mathcal{P}$. Now suppose the approximate loss functions are also piecewise constant with pieces $\tilde{A}_t^{(1)}, \ldots, \tilde{A}_t^{(M)}$ that partition $\mathcal{P}$, and she receives information about the constant piece $\tilde{A}_t(\rho_t)$ containing the played point by receiving the ends points of interval $\tilde{A}_t$ and approximate loss value $\tilde{l}_t$ for the observed piece $\tilde{A}_t$ with $|\tilde{l}_t - l_t| \leq \gamma$ for most of the interval $\tilde{A}_t$, except possibly finitely many small intervals with total length $\epsilon$, where $l_t$ is the true loss function. This satisfies the definition of $(\epsilon, \gamma)$-approximate semi-bandit feedback. See Figure 1 for an illustration. This simple example captures the semi-supervised loss $l_{A(G(\rho),L,U)}$ (where in fact the true loss function is also piecewise constant [Balcan and Sharma, 2021]), but our analysis in this section applies to more general *piecewise-Lipschitz* loss functions, and for high dimensional Euclidean action space. This approximate feedback model generalizes the "exact" semibandit feedback model of Balcan et al. [2020b] (which in turn generalizes the standard 'full information' setting that corresponds to $M = 1$) and is useful for cases where computing the exact feedback set or loss function is infeasible or computationally expensive. Our model also generalizes the approximate loss functions of Balcan et al. [2020e] where positive results (data-dependent generalization guarantees) are shown for $(0, \gamma)$-approximate full-information ($M = 1$) feedback in the distributional setting. This extension is crucial for applying our techniques of efficient graph learning by computing approximate loss values for the learned graph.

We give a general online learning algorithm in the presence of approximate semi-bandit feedback, and we show that our algorithm achieves sub-linear regret bounds. In particular, our results indicate how the approximation in the loss function impacts the regret of our algorithm and provides a way to quantify the accuracy-efficiency trade-off (better loss approximation can improve regret in Theorem 3.1, but at the cost of efficiency in Theorems 5.1, 5.2).

**Algorithm 1** APPROXIMATE CONTINUOUS EXP3-SET($\lambda$)

**Input:** step size $\lambda \in [0, 1]$.
Initialize $w_1(\rho) = 1$ for all $\rho \in \mathcal{P}$.
**for** $t = 1, \ldots, T$ **do**
  Sample $\rho_t$ according to $p_t(\rho) = \frac{w_t(\rho)}{\int_{\mathcal{P}} w_t(\rho) d\rho}$.
  Play $\rho_t$ and suffer loss $l_t(\rho_t)$.
  Observe $(\gamma, \epsilon)$-approximate feedback $\tilde{l}_t(\rho)$ over set $\tilde{A}_t$ with $\rho_t \in \tilde{A}_t$
  Update $w_{t+1}(\rho) = w_t(\rho) \exp(-\lambda \hat{l}_t(\rho))$, where $\hat{l}_t(\rho) = \frac{\mathbf{I}\{\rho \in \tilde{A}_t\}}{\int_{\tilde{A}_t} p_t(\rho) d\rho} \tilde{l}_t(\rho)$.

**Theorem 3.1.** *Suppose $l_1, \ldots, l_T : \mathcal{P} \to [0, 1]$ is a sequence of $\beta$-dispersed loss functions, and the domain $\mathcal{P} \subset \mathbb{R}^d$ is contained in a ball of radius $R$. The Approximate Continuous Exp3-Set algorithm (Algorithm 1) achieves expected regret $\tilde{O}(\sqrt{dMT \log(RT)} + T^{1-\min\{\beta,\beta'\}})$ with access to $(\epsilon, \gamma)$-approximate semi-bandit feedback with system size $M$, provided $\gamma \leq T^{-\beta'}, \epsilon \leq \mathsf{vol}(\mathcal{B}(T^{-\beta}))T^{-\beta'}$, where $\mathcal{B}(r)$ is a $d$-ball of radius $r$.*

*Proof Sketch.* We adapt the CONTINUOUS-EXP3-SET analysis of Alon et al. [2017], Balcan et al. [2020b]. Define weights $w_t(\rho)$ over the parameter space $\mathcal{P}$ as $w_1(\rho) = 1$ and $w_{t+1}(\rho) = w_t(\rho) \exp(-\eta \hat{l}_t(\rho))$ and normalized weights $W_t = \int_{\mathcal{P}} w_t(\rho) d\rho$. Note that $p_t(\rho) = \frac{w_t(\rho)}{W_t}$. We give upper and lower bounds on the quantity $\mathbb{E}[\log W_{T+1}/W_1]$, i.e. the expected value of the log-ratio of normalized weights, and bound the slackness induced in these bounds due to $(\epsilon, \gamma)$-approximate feedback. Our analysis shows that, provided the error terms $\epsilon, \gamma$ are sub-constant in $T$ as stated, we achieve sublinear expected regret. □

In Theorem 3.1. $\beta'$ measures the net impact of approximate feedback on the regret of Algorithm 1. In particular, it shows that approximation can affect regret when ($\gamma, \epsilon$ are such that) $\beta' < \beta$ and $\beta' < \frac{1}{2}$. The bound in Theorem 3.1 is good for sufficiently small $\gamma, \epsilon$. However, very small $\gamma, \epsilon$ can come at the expense of speed. In more detail, our results in Section 5 discuss how approximate feedback can be algorithmically implemented and useful to obtain faster runtime (runtime bounds are weaker for smaller $\epsilon$). Together, the results quantify an accuracy-efficiency trade-off, and indicate how to set the approximation parameters to improve the efficiency (of graph hyperparameter tuning) without sacrificing the accuracy.

# 4 LEARNING SPARSE GRAPH FAMILIES

Using (weighted) edges for the $k$-nearest neighbors to use a sparse graph is well-known as an optimization for computational efficiency in semi-supervised learning [Delalleau et al., 2005, Wang et al., 2016]. Here we will show that it also formally reduces the learning theoretic complexity, for the problem of graph hyperparameter tuning. Proofs from this section appear in Appendix A.

We can upper bound the pseudodimension of the class of loss functions for sparse graph families, where only $k$ nearest neighbors are connected, for tunable parameter $k \leq K$. This upper bound improves on the $O(n)$ bound from [Balcan and Sharma, 2021] since $K \leq n$, and involves a more careful argument to bound the number of possible label patterns.

**Theorem 4.1.** *The pseudo-dimension of $\mathcal{H}_{k,\sigma}$ is $O(K + \log n)$ when the labeling algorithm $A$ is the mincut approach of Blum and Chawla [2001].*

The above argument gives a better sample complexity than dense graphs, for which the pseudo-dimension is known to be $\Theta(n)$ [Balcan and Sharma, 2021]. We can also give upper bounds on the pseudo-dimension for $H_{k,r}$, the $k$-nearest neighbor graph that adds edges only in $r$-neighborhood, which implies existence of sample and computationally efficient algorithms for learning the best graph parameter $\rho = (k, r)$ using standard results.

**Theorem 4.2.** *The pseudo-dimension of $\mathcal{H}_{k,r}$ is $O(\log n)$ for any labeling algorithm $A$.*

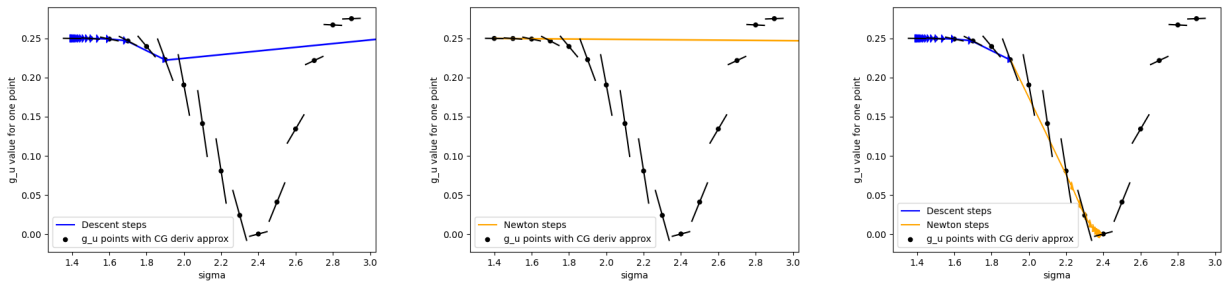
Note that the lower bounds from Balcan and Sharma [2021] imply that the above bound is asymptotically tight. Our bounds in this section imply upper bounds on number of problem instances needed for learning the best parameter values for the respective graph families [Balcan, 2020] in the distributional setting. More precisely, we can bound the sample complexity of $(\epsilon, \delta)$-uniformly learning (Appendix A.1).

**Theorem 4.3** (Anthony and Bartlett [1999])**.** *Suppose $\mathcal{H}$ is a class of functions $\mathcal{X} \to [0, 1]$ having pseudo-dimension* $\text{PDIM}(\mathcal{H})$. *For every $\epsilon > 0, \delta \in (0, 1)$, the sample complexity of $(\epsilon, \delta)$-uniformly learning the class $\mathcal{H}$ is $O\left(\frac{1}{\epsilon^2}\left(\text{PDIM}(\mathcal{H}) \ln \frac{1}{\epsilon} + \ln \frac{1}{\delta}\right)\right)$.*

# 5 SCALABILITY WITH APPROXIMATION GUARANTEES

We will now present and analyse an algorithm (Algorithm 3) for computing approximate semi-bandit feedback for the dual semi-supervised loss $l(\sigma)$ over $\sigma \in [\sigma_{\min}, \sigma_{\max}]$ (we assume number of nearest-neighbors $k$ is a fixed constant in the following), where $\sigma$ is the Gaussian bandwidth parameter (Def. 1). Our algorithm is a scalable version of Algorithm 4 of Balcan and Sharma [2021]. Our proposed approach involves two main modifications noted below.

- Our Algorithm 3 uses approximate soft labels $f(\sigma)_\epsilon$ and gradients $\frac{\partial f}{\partial \sigma}_\epsilon$. We use the *conjugate gradient* method to compute these approximations, and provide implementations for the harmonic objective minimization approach




(a) Gradient Descent Only (b) Newton's Method Only (c) Our Method

Figure 2: An instance of a node $u$ for graph $G$ on a subset of the MNIST dataset, where finding local minima of $g_u(\sigma) = (f_u(\sigma) - \frac{1}{2})^2$ is challenging for both Gradient descent and Newton steps.

of Zhu et al. [2003], as well as the efficient algorithm of Delalleau et al. [2005] with time complexity bounds (Algorithm 2 and DBLR05APPROX resp. below).

- We use the approximate gradients to locate points where $f(\sigma^*) = \frac{1}{2}$, corresponding to $\sigma$ value where the predicted label flips. We use these points to find $(\epsilon, \epsilon)$-approximate feedback sets. We propose the use of smaller of Newton's step and gradient descent for better convergence to these points (line 10 in Algorithm 3; Balcan and Sharma [2021] use only Newton's method). We motivate this step by giving an example (from a real dataset) where the gradients are both too small and too large near the minima (Figure 2). This makes convergence challenging for both gradient descent and Newton's method, but the combination is effective even in this setting. We also give convergence guarantees and runtime bounds for Algorithm 3 in the presence of approximate gradients (Theorems 5.1, 5.2).

We first describe how to instantiate the sub-routine $\mathcal{A}$ to compute approximate soft labels in Algorithm 3 (full details in Appendix B for the interested reader).

Algorithm 2 computes the soft label that optimizes the harmonic function objective Zhu et al. [2003] and gradient for a given value of graph parameter $\sigma$ for a fixed unlabeled node $u$. This is accomplished by running the conjugate gradient for given number of iterations to solve systems corresponding to the harmonic function objective and its gradient.

(Informal) DBLR05APPROX$(G, f_L, i, \sigma, \epsilon)[\tilde{U}, \lambda]$: This algorithm computes the soft label and gradient corresponding to the efficient algorithm of Delalleau et al. [2005] for a graph $G$ with parameter $\sigma$ for a fixed unlabeled node $i \in U$. Unlike Algorithm 2, a matrix inverse is approximated via iterations of the CG method for the Laplacian of a small subset of unlabeled 'training' nodes $\tilde{U} \subset U$ along with a set of labeled nodes $L$. The labels of $i \in U \setminus \tilde{U}$ ('testing' nodes) are determined by summing the labels of each $x \in \tilde{U} \cup L$, weighted by $W_{ij}(x, i)$. The full algorithm is presented as Algorithm 1 in section B.3 of the appendix.

**Algorithm 2** HARMONICAPPROXIMATION$(G, f_L, u, \sigma, \epsilon)$

1: **Input:** Graph $G$ with labeled nodes $f_L$, unlabeled node $u$, query parameter $\sigma$, error tolerance $\epsilon$.
2: **Output:** approximate soft label $f_{u,\epsilon}$ and approximate gradient $\frac{\partial f_u}{\partial \sigma}_\epsilon$.
3: Let CG$(A, b, t)$ represent running the conjugate gradient method for $t$ iterations to solve $Ax = b$.
4: Let $t_\epsilon$ indicate the number of iterations sufficient for $\epsilon$-approximation of $f_u(\sigma)\frac{\partial f_u}{\partial \sigma}$ (Theorem B.2 in the appendix).
5: Let $f_{U,\epsilon}(\sigma) = \text{CG}\left((I - P_{UU}), P_{UL} f_L, t_\epsilon\right)$, where $D_{ij} := \mathbb{I}[i = j] \sum_k W_{ik}, P = D^{-1}W$.
6: Let $\frac{\partial f}{\partial \sigma}_\epsilon = \text{CG}\big((I - P_{UU}), \big(\frac{\partial P_{UU}}{\partial \sigma} f_{U,\epsilon} + \frac{\partial P_{UL}}{\partial \sigma} f_L\big), t_\epsilon\big)$, where

$$\frac{\partial P_{ij}}{\partial \sigma} = \frac{\frac{\partial w(i,j)}{\partial \sigma} - P_{ij} \sum_{k \in L+U} \frac{\partial w(i,k)}{\partial \sigma}}{\sum_{k \in L+U} w(i,k)},$$

$$\frac{\partial w(i,j)}{\partial \sigma} = \frac{2w(i,j)d(i,j)^2}{\sigma^3}.$$

7: **return** $f_{u,\epsilon}(\sigma), \frac{\partial f_u}{\partial \sigma}_\epsilon$.

The algorithm finds an $\epsilon$ approximation of $\tilde{f}_u(\sigma) \cdot \frac{\partial \tilde{f}_u}{\partial \sigma}$ using $O\left(\sqrt{\kappa(A)} \log\left(\frac{\lambda(|L_{\text{Labels}}| + |\tilde{U}_{\text{Labels}}|)}{\epsilon \sigma_{\min} \lambda_{\min}(A)}\right)\right)$ conjugate gradient iterations, where $\kappa$ is the condition number, $\tilde{U} \subset U$ is a small subset, and $\lambda$ is a hyperparameter. Formal statement and proof is in the appendix (Theorem B.3).

Our main result is the following guarantee on the performance of Algorithm 3, which captures the approximation-efficiency trade-off for the algorithm. Compared to the $\tilde{O}(n^4)$ running time of the approach in Balcan and Sharma [2021], our algorithm runs in time $\tilde{O}(n^2)$ for sparse kNN graphs (i.e. $k$-nearest neighbors with small constant $k$). To achieve this speedup, we replace an $O(n^3)$ matrix inverse for a given unlabeled point with a fixed number of Conjugate

**Algorithm 3** APPROXFEEDBACKSET$(G, f_L, \sigma_0, \epsilon, \eta, \mathcal{A})$

**Input:** Graph $G$ with unlabeled nodes $U$, labels $f_L$, query parameter $\sigma_0$, error tolerance $\epsilon$, learning rate $\eta$, algorithm $\mathcal{A}$ to estimate soft labels and derivatives at any $\sigma$ (e.g. Algorithm 2).
**Output:** Estimates for piecewise constant interval containing $\sigma_0$, and function value at $\sigma$.
Initialize $\sigma_l = \sigma_h = \sigma_0$.
**for all** $u \in U$ **do**
  Initialize $n = 0, \lambda_0 = 1, y_0 = \sigma_0$.
  **while** $|\sigma_{n+1} - \sigma_n| \geq \epsilon$ **do**
    Compute $f_{u,\epsilon}(\sigma), \frac{\partial f_u}{\partial \sigma}_{\epsilon}$ as $\mathcal{A}(G, f_L, u, \sigma_n, \epsilon)$
    Set $g_u(\sigma_n) = (f_{u,\epsilon}(\sigma_n) - \frac{1}{2})^2$, $g'_u(\sigma_n) = 2\left(f_{u,\epsilon}(\sigma_n) - \frac{1}{2}\right)\left(\frac{\partial f_u}{\partial \sigma}_{\epsilon}\right)$.
    $\xi_{\text{GD}} \leftarrow \eta g'_u(\sigma_n)$; $\xi_{\text{Newton}} \leftarrow 2\frac{g_u(\sigma_n)}{g'_u(\sigma_n)}$.
    $y_{n+1} = \sigma_n - \min\{\xi_{\text{GD}}, \xi_{\text{Newton}}\}$.
    **if** $\xi_{\text{GD}} < \xi_{\text{Newton}}$ **then**
      $\lambda_{n+1} = \frac{1+\sqrt{1+4\lambda_n^2}}{2}, \gamma_n = \frac{1-\lambda_n}{\lambda_{n+1}}, \sigma_{n+1} = (1-\gamma_n)y_{n+1} + \gamma_n y_n$
    **else**
      $\sigma_{n+1} = y_{n+1}$
    $n \leftarrow n+1$
  $\sigma_l = \min\{\sigma_l, \sigma_{n+1}\}, \sigma_h = \max\{\sigma_h, \sigma_{n+1}\}$.
**return** $[\sigma_l, \sigma_h], f_\epsilon(\sigma_0)$.

Gradient iterations taking time $O(|E_G|)$, where $|E_G|$ is the number of edges for graph $G$ corresponding to the matrix being inverted. Combined with our general algorithm for approximate data-driven algorithm design (Theorem 3.1), we obtain $\tilde{O}(\sqrt{T})$ expected regret for online graph parameter tuning with approximate semi-bandit feedback, provided we run Algorithm 3 with $\epsilon \leq \frac{1}{\sqrt{T}}$. For our proof, we will assume that the soft label function $f_u(\sigma)$ is convex and smooth (i.e. derivative w.r.t. $\sigma$ is Lipschitz continuous) for estimating the convergence rates. In Section 6, we observe that our algorithm works well in practice even when these assumptions on $f$ are not satisfied, and it would be interesting to extend our analysis to weaker assumptions on the soft label.

**Theorem 5.1.** *Using Algorithm 2 for computing $\epsilon$-approximate soft labels and gradients for the harmonic objective of Zhu et al. [2003], if $f_u(\sigma)$ is convex and smooth, Algorithm 3 computes $(\epsilon, \epsilon)$-approximate semi-bandit feedback for the semi-supervised loss $l(\sigma)$ in time $O\Big(|E_G| n\sqrt{\kappa(\mathcal{L}_{UU})}\log\Big(\frac{n\Delta}{\epsilon\lambda_{\min}(\mathcal{L}_{UU})}\Big)\log\log\frac{1}{\epsilon}\Big)$, where $|E_G|$ is the number of edges in graph $G$, $\mathcal{L}_{UU} = I - P_{UU}$ is the normalized grounded graph Laplacian (with labeled nodes grounded), $\Delta = \sigma_{\max} - \sigma_{\min}$ is the size of the parameter range and $\kappa(M) = \frac{\lambda_{\max}(M)}{\lambda_{\min}(M)}$ denotes the condition number of matrix $M$.*

*Proof Sketch.* As noted in Balcan and Sharma [2021], the loss $l(\sigma)$ is discontinuous at $\sigma^*$ only if $f_u(\sigma^*) = \frac{1}{2}$. Algorithm 3 finds these critical points by finding roots/zeros of $\left(f_u(\sigma) - \frac{1}{2}\right)^2$. We show (Theorem C.1 in the appendix) that if $f$ is convex and smooth, Nesterov's accelerated descent [Nesterov, 1983] quadratically converges to within $\epsilon$ of such roots, given an $O(\frac{\epsilon}{\Delta})$-approximations of $f\frac{\partial f}{\partial \sigma}$ and $\left|\frac{\partial f}{\partial \sigma}\right| < \frac{1}{\epsilon\lambda_{\min}(G_A)}$. Newton's method converges quadratically to within $\epsilon$, given $\epsilon$-approximations of $f\frac{\partial f}{\partial \sigma}$ (Theorem C.2 in the appendix). We use Algorithm 2 to find suitable $\epsilon$-approximations of $f\frac{\partial f}{\partial \sigma}$ in time $O\left(\sqrt{\kappa(\mathcal{L}_{UU})}\log\left(\frac{n\Delta}{\epsilon\lambda_{\min}(\mathcal{L}_{UU})}\right)\right)$ (Theorem B.2). We argue that if the derivative $\frac{\partial f}{\partial \sigma}$ is large (i.e. the condition on $\frac{\partial f}{\partial \sigma}$ for Theorem C.2 does not hold), then the Newton step will be less than $\epsilon$. Since the algorithm uses the smaller of the Newton and Nesterov updates, Algorithm 3 will terminate for given $u \in U$. By quadratic convergence, we need $O(\log\log\frac{1}{\epsilon})$ iterations in Algorithm 3 for each of the $O(n)$ unlabeled elements. Finally, noting that the Conjugate Gradient method takes $O(|E_G|)$ time per iteration, we obtain the claimed bound on runtime. □

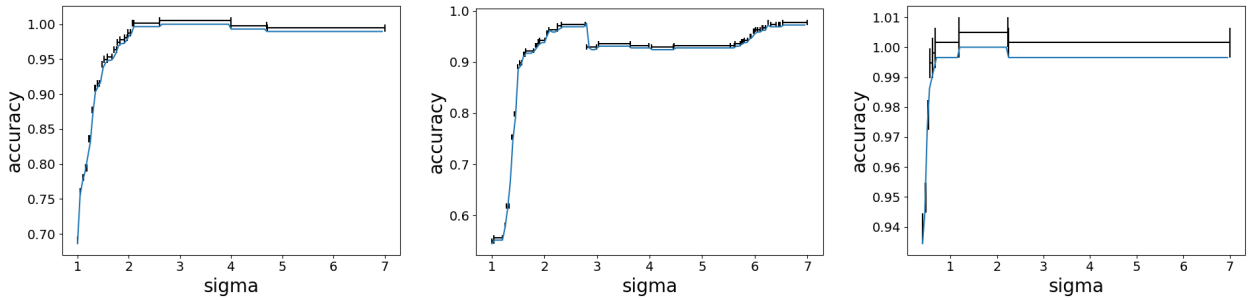
Above analysis can be adapted to obtain the following guarantee for tuning $\sigma$ in the efficient algorithm of Delalleau et al. [2005]. While the above result guarantees a running time of $\tilde{O}(n^2)$ for kNN graphs, learning the graph can be done even more efficiently for the scalable approach of Delalleau et al. [2005]. Their algorithm minimizes a proxy for the harmonic objective given by

$$\frac{1}{2}\sum_{u,v\in\tilde{U}} w(u,v)(f(u) - f(v))^2 + \lambda\sum_{w\in L}(f(w) - y_w)^2$$

where $\tilde{U} \subset U$ and $\lambda$ are hyperparameters. In particular, one chooses a small set $\tilde{U}$ with $|\tilde{U}| \ll n$ and efficiently extrapolates the harmonic labels on $\tilde{U}$ to the rest of $U$ using a Parzen windows based extrapolation. As before, the success of this more efficient approach also depends on the choice of the graph $G$ used. Our Algorithm 3 obtains good theoretical guarantees in this case as well, with appropriate choice of algorithm $\mathcal{A}$ (namely DBLR05APPROX).

**Theorem 5.2.** *(Informal) Given an algorithm for computing $\epsilon$-approximate soft labels and gradients for the efficient semi-supervised learning algorithm of Delalleau et al. [2005] (*DBLR05APPROX*), Algorithm 3 computes $(\epsilon, \epsilon)$-approximate semi-bandit feedback for the semi-supervised loss $l(\sigma)$ in time $O\Big(|E_{G_{\tilde{U}}}| n\sqrt{\kappa(\mathcal{L}_A)}\log\Big(\frac{\lambda(|L_{Labels}|+|\tilde{U}_{Labels}|)\Delta}{\epsilon\sigma_{\min}\lambda_{\min}(\mathcal{L}_A)}\Big)\log\log\frac{1}{\epsilon}\Big)$, where all values are defined as in* DBLR05APPROX

*Proof Sketch.* The proof follows in the same manner as Theorem 5.1, except we now use DBLR05APPROX to bound




(a) MNIST (b) Fashion-MNIST (c) USPS

Figure 3: Loss intervals calculated with approximate soft-labels via Algorithm 2, kNN = 6, $|U| = 300$. Blue line corresponds to true loss, black intervals are estimated constant loss intervals.

the number of iterations of the CG method. □

We empirically observe (Figure 5, Appendix E) that sparsity (using only kNN edges, and nodes in $\tilde{U}$) results in well-conditioned matrices (bounded $\sqrt{\kappa(A)}$) in the considered parameter range $[\sigma_{\min}, \sigma_{\max}]$.

**Remark 1.** *In this work we have focused on efficient graph learning for the harmonic objective approach of Zhu et al. [2003] and the efficient algorithm of Delalleau et al. [2005]. Developing approaches that work for other efficient algorithms from the literature [Sinha and Belkin, 2009, Wang et al., 2016] constitutes interesting future work.*

# 6 EXPERIMENTS

In this section we evaluate computational speedups as a result of using the conjugate gradient method and implement Algorithm 3 to compute pieces of the loss function under different labeling techniques.

*Setup*: We consider the task of semi-supervised binary classification (classes 0 and 1) on image datasets. As in Delalleau et al. [2005], we pre-process data instances via Principal Component Analysis, keeping the first 45 principal components. We measure *distance* between any pairs of images by $L_2$ distance between principal components, and set weights via Gaussian Kernel parameterized by $\sigma$. When testing computational speedup using the CG method, we draw random subsets of the full dataset at varying sizes of $n$, with labeled set size $L = \frac{n}{10}$. When computing approximate matrix inverses, we use $t = 20$ conjugate gradient iterations.

*Datasets*: We use three established benchmark image datasets – MNIST, Fashion-MNIST, and USPS. Both the MNIST dataset (handwritten digits, Lecun et al. [1998]) and the FashionMNIST dataset (mock fashion items, Xiao et al. [2017]) consist of 28 by 28 grayscale images with 10 classes, and 6000 images per class. The US Postal Service (USPS) dataset [Hull, 1994] has 7291 handwritten digits downscaled to 16 by 16 grayscale images. For MNIST and USPS, binary classification between classes 0 and 1 corresonds to classifying between handwritten 0s and 1s. For FashionMNIST, it corresponds to classifying between classes T_shirt and Trouser.

## 6.1 EFFICIENT FEEDBACK SET COMPUTATION (ALGORITHM 3)

We consider the problem of finding approximate intervals of the piecewise constant loss $l(\sigma)$ using Algorithm 2 with the number of unlabeled points $n \in \{100, 300, 500\}$. By finding a set of these piecewise constant componenets, we are able to search the continuous paramter space exhaustively, with an optimal hyperparameter being any parameter in the loss interval with lowest loss. We do this for both the complete graph, as well as kNN with $k = 6$, setting number of labeled examples $|L| = 10$. We do the same with DBLR05APPROX as algorithm $\mathcal{A}$, with (uniformly random) subset $\tilde{U}$ size 50 and hyperparameter $\lambda = 1.4$.

We motivate design choices in Algorithm 3 by examining $(\epsilon, \epsilon)$-approximate semi-bandit feedback for the semi-supervised loss $l(\sigma)$ produced by the algorithm. For full implementation details, see Appendix E.1.

*Results:* Figure 3 (as well as Fig. 2, 3, and 4 in Appendix) indicates that the CG method can be used to find accurate piecewise intervals on real loss functions. We see over 10x speedup for using the CG method as opposed to using matrix inversion for computing soft labels via Algorithm 2 (Table 1). This is due to the speedup in inversion time between a full matrix inverse and the CG method. For specific times for these two inversion techniques, see Tables 1-4 in Appendix. When using DBLR05APPROX to compute labels, we see a speedup of 100x with slower asymptotic increase over time in comparison to the full inverse using Algorithm 2. Here the speedup is due to the fact that the CG method is only run for size 50 subsets of $U$, with $O(|\tilde{U}|)$ algebraic steps afterwards

Table 1: Time (in seconds) per Interval (TpI) and Number of Intervals ($M$) using both the Conjugate Gradient method (CG, $t = 20$ iterations) and Matrix Inverse (MI) on full graphs or kNN, $k = 6$ graphs using Algorithm 3. Approximate soft labels are computed using one of Algorithm 2, DBLR05APPROX.

| Dataset | Size | Algorithm 2 | | | | Algorithm 2 (kNN) | | | | DBLR05APPROX (kNN) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | TpI | | $M$ | | TpI | | $M$ | | TpI | $M$ |
| | | CG | MI | CG | MI | CG | MI | CG | MI | CG | |
| MNIST | 100 | **1.50** | 3.22 | 38.9 | 26.0 | 6.33 | 10.11 | 17.36 | 5.7 | 2.22 | 11.5 |
| | 300 | 15.87 | 346.46 | 22.1 | 26.8 | 19.98 | 2405.70 | 22.1 | 7.9 | **5.98** | 26.2 |
| | 500 | 57.90 | 818.11 | 26.6 | 23.6 | 20.99 | 6791.79 | 26.6 | 7.9 | **7.60** | 36.1 |
| | 12615 | - | - | - | - | - | - | - | - | 46.36 | - |
| Fashion-MNIST | 100 | **1.79** | 3.56 | 39.0 | 23.9 | 11.79 | 12.49 | 18.55 | 8.3 | 2.91 | 12.7 |
| | 300 | 11.73 | 268.20 | 45.6 | 38.9 | 21.08 | 1447.56 | 35.9 | 21.2 | **7.24** | 33.6 |
| | 500 | 39.98 | 766.73 | 50.9 | 37.6 | 35.91 | 6311.03 | 38.7 | 29.0 | **9.21** | 44.6 |
| | 11950 | - | - | - | - | - | - | - | - | 32.3 | - |
| USPS | 100 | **1.58** | 3.47 | 25.4 | 18.6 | 6.91 | 12.53 | 4.7 | 1.3 | 2.12 | 6.67 |
| | 300 | 16.63 | 238.16 | 30.1 | 18.8 | 29.86 | 68.70 | 5.6 | 1.2 | **6.34** | 16.14 |
| | 500 | 57.50 | 755.94 | 39.1 | 17.8 | 63.18 | 31.54 | 6.1 | 1.0 | **12.37** | 20.4 |
| | 2149 | - | - | - | - | - | - | - | - | 27.28 | - |

to find soft labels/gradients for a given unlabeled point in $U$. We find that the calculation of feedback sets in kNN graphs takes longer to find a single interval on smaller data subsets, but the runtime asymptotically grows slower than for complete graphs. This is likely due to longer piecewise constant intervals for kNN graphs, leading to a larger amount of matrix inverse calculations performed per interval (consistent with $M$ values in Table 1). We also find that the CG method is more robust to higher condition number than the full inverse for low $\sigma$ values, indicating that the CG method is not only faster, but also can be more stable than full matrix inversion for ill-conditioned grounded Laplacians.

**CG Method Details.** We run $t$ iterations of the CG method to approximate grounded Laplacian inverses for finding soft labels in Algorithm 2. We consider $t \in \{5, 10, 20\}, \sigma' \in [1, 7]$ when finding optimal values in terms of unlabeled classification accuracy. We use SciPy [Virtanen et al., 2020] for performing the conjugate gradient method as well as storing nearest-neighbor matrices sparsely for time speedup. For all three datasets, we find parameters to closely match/surpass the performance of the harmonic solution with matrix inverse (i.e. prior work) for optimal $\sigma$ in both the complete graph and kNN setting with an order of magnitude or more speedup (Tables 1 and 2 in the appendix). We find that there is little time difference between $t = 5, 15$ and 20 conjugate gradient iterations. We also find slight speedup for kNN graphs.

# 7 CONCLUSION

We provide a general analysis for approximate data-driven parameter tuning in the presence of approximate feedback. We show how this approximate feedback may be efficiently implemented for learning the graph in semi-supervised learning using the conjugate gradient method. We further show the significance of using sparse nearest neighborhood graphs for semi-supervised learning – formally they need provably fewer samples for learning compared to using the complete graph, and moreover, in practice, they lead to better conditioned matrices for which our approach converges faster. We quantify the efficiency vs. accuracy trade-off for our approach, and empirically demonstrate its usefulness in more efficiently learning the graph for classic harmonic objective based algorithms [Zhu et al., 2003, Delalleau et al., 2005]. We believe this is an important step in making the data-driven approach practical for semi-supervised learning, and would potentially be useful for making data-driven algorithm design more useful for other problems. Interesting future directions include extending this approach to learning the graph for modern graph neural network based methods (which still assume a given graph) for semi-supervised learning, and applications to other parameter tuning problems where exact feedback may be computationally expensive to obtain.

# 8 ACKNOWLEDGEMENT

We thank Nina Balcan for suggesting the problem and for useful discussions. We also thank David Tang and Advait Nene for useful discussions.

## References

Noga Alon, Nicolo Cesa-Bianchi, Claudio Gentile, Shie Mannor, Yishay Mansour, and Ohad Shamir. Nonstochastic multi-armed bandits with graph-structured feedback. *SIAM Journal on Computing*, 46(6):1785–1826, 2017.

Martin Anthony and Peter L Bartlett. *Neural network learning: theoretical foundations*, volume 9. Cambridge University Press, Cambridge, 1999.

O. Axelsson. A class of iterative methods for finite element equations. *Computer Methods in Applied Mechanics and Engineering*, 9(2):123–137, 1976. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(76)90056-6. URL https://www.sciencedirect.com/science/article/pii/0045782576900566.

Maria-Florina Balcan. Book chapter Data-Driven Algorithm Design. In *Beyond Worst Case Analysis of Algorithms, T. Roughgarden (Ed)*. Cambridge University Press, 2020.

Maria-Florina Balcan and Avrim Blum. A discriminative model for semi-supervised learning. *Journal of the ACM (JACM)*, 57(3):1–46, 2010.

Maria-Florina Balcan and Dravyansh Sharma. Data driven semi-supervised learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. In *Conference on Learning Theory*, pages 213–274. PMLR, 2017.

Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *International conference on machine learning*, pages 344–353. PMLR, 2018a.

Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Dispersion for data-driven algorithm design, online learning, and private optimization. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 603–614. IEEE, 2018b.

Maria-Florina Balcan, Travis Dick, and Colin White. Data-driven clustering via parameterized lloyd's families. *Advances in Neural Information Processing Systems*, 31: 10641–10651, 2018c.

Maria-Florina Balcan, Avrim Blum, Dravyansh Sharma, and Hongyang Zhang. On the power of abstention and data-driven decision making for adversarial robustness. *arXiv preprint arXiv:2010.06154*, 2020a.

Maria-Florina Balcan, Travis Dick, and Wesley Pegden. Semi-bandit optimization in the dispersed setting. In *Conference on Uncertainty in Artificial Intelligence*, pages 909–918. PMLR, 2020b.

Maria-Florina Balcan, Travis Dick, and Dravyansh Sharma. Learning piecewise Lipschitz functions in changing environments. In *International Conference on Artificial Intelligence and Statistics*, pages 3567–3577. PMLR, 2020c.

Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. Learning to optimize computational resources: Frugal training with generalization guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3227–3234, 2020d.

Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. Refined bounds for algorithm configuration: The knife-edge of dual class approximability. In *International Conference on Machine Learning*, pages 580–590. PMLR, 2020e.

Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *International Conference on Machine Learning (ICML)*, 2001.

Olivier Delalleau, Yoshua Bengio, and Nicolas Le Roux. Efficient non-parametric function induction in semi-supervised learning. In *International Workshop on Artificial Intelligence and Statistics*, pages 96–103. PMLR, 2005.

Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017.

William W Hager and Hongchao Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

Magnus R Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving. *Journal of research of the National Bureau of Standards*, 49(6):409, 1952.

J.J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994. doi: 10.1109/34.291440.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Senthil Kumar, Leman Akoglu, Nitesh Chawla, Saurabh Nagrecha, Vidyut M Naware, Tanveer Faruquie, and Hays McCormick. Kdd workshop on machine learning in finance. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4882–4883, 2022.

Pedro Larrañaga, David Atienza, Javier Diaz-Rozo, Alberto Ogbechie, Carlos Puerto-Santana, and Concha Bielza. *Industrial applications of machine learning*. CRC press, 2018.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yu E Nesterov. A method for solving the convex programming problem with convergence rate $o(\frac{1}{k^2})$. In *Dokl. Akad. Nauk SSSR,*, volume 269, pages 543–547, 1983.

Kohei Ozaki, Masashi Shimbo, Mamoru Komachi, and Yuji Matsumoto. Using the mutual k-nearest neighbor graphs for semi-supervised classification on natural language data. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 154–162, 2011.

David Pollard. *Convergence of stochastic processes*. Springer Science & Business Media, 2012.

Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. *Advances in Neural Information Processing Systems*, 30, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

Furao Shen, Jing Chao, and Jinxi Zhao. Forecasting exchange rate using deep belief networks and conjugate gradient method. *Neurocomputing*, 167:243–253, 2015. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2015.04.071. URL https://www.sciencedirect.com/science/article/pii/S0925231215005408.

Kaushik Sinha and Mikhail Belkin. Semi-supervised learning using sparse eigenfunction bases. *Advances in Neural Information Processing Systems*, 22, 2009.

Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. Applications of machine learning in drug discovery and development. *Nature reviews Drug discovery*, 18(6):463–477, 2019.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Nisheeth K Vishnoi. Laplacian solvers and their algorithmic applications. *Theoretical Computer Science*, 8(1-2):1–141, 2012.

Meng Wang, Weijie Fu, Shijie Hao, Dacheng Tao, and Xindong Wu. Scalable semi-supervised learning by efficient anchor graph regularization. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1864–1877, 2016.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xiaojin Zhu. *Semi-supervised learning with graphs*. Carnegie Mellon University, 2005.

Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.

Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 912–919, 2003.