Projection-free Graph-based Classifier Learning using Gershgorin Disc Perfect Alignment

Anonymous Author(s)
Affiliation
Address
email

Abstract

In semi-supervised graph-based binary classifier learning, a subset of known labels $\hat{x}_i$ are used to infer unknown labels, assuming that the label signal $\mathbf{x}$ is smooth with respect to a similarity graph specified by a Laplacian matrix. When restricting labels x_i to binary values, the problem is NP-hard. While a conventional semi-definite programming (SDP) relaxation can be solved in polynomial time using, for example, the alternating direction method of multipliers (ADMM), the complexity of iteratively projecting a candidate matrix $\mathbf{M}$ onto the positive semi-definite (PSD) cone ($\mathbf{M} \succeq 0$) remains high. In this paper, leveraging a recent linear algebraic theory called Gershgorin disc perfect alignment (GDPA), we propose a fast projection-free method by solving a sequence of linear programs (LP) instead. Specifically, we first recast the SDP relaxation to its SDP dual, where a feasible solution $\mathbf{H} \succeq 0$ can be interpreted as a Laplacian matrix corresponding to a balanced signed graph sans the last node. To achieve graph balance, we split the last node into two that respectively contain the original positive and negative edges, resulting in a new Laplacian $\bar{\mathbf{H}}$. We repose the SDP dual for solution $\bar{\mathbf{H}}$, then replace the PSD cone constraint $\bar{\mathbf{H}} \succeq 0$ with linear constraints derived from GDPA—sufficient conditions to ensure $\bar{\mathbf{H}}$ is PSD—so that the optimization becomes an LP per iteration. Finally, we extract predicted labels from our converged LP solution $\bar{\mathbf{H}}$. Experiments show that our algorithm enjoyed a $40\times$ speedup on average over the next fastest scheme while retaining comparable label prediction performance.

1 Introduction

Binary classification—assignment of labels to an N-sample set $\mathbf{x} \in \{-1, 1\}^N$ to separate two distinct classes—is a basic machine learning problem [1]. One common setting is semi-supervised graph classifier learning, where M known labels, $\hat{x}_i, 1 \leq i \leq M$, are used to infer $N - M$ unknown labels x_i, $M + 1 \leq i \leq N$, in signal $\mathbf{x}$, assuming that $\mathbf{x}$ is smooth with respect to (w.r.t.) a similarity graph $\mathcal{G}$ specified by a graph Laplacian matrix $\mathbf{L}$ [2, 3, 4]. This graph-based binary classification problem is NP-hard in general [5]. A conventional *semi-definite programming* (SDP) relaxation [6] replaces the binary label constraint with a more relaxed *positive semi-definite* (PSD) cone constraint (*i.e.*, matrix variable $\mathbf{M}$ related to $\mathbf{x}\mathbf{x}^\top$ satisfying $\mathbf{M} \succeq 0$), and the relaxed problem can be solved in polynomial time using, for example, the *alternating direction method of multipliers* (ADMM) [7]. However, ADMM still requires projection to the PSD cone $\mathcal{S} = \{\mathbf{M} \,|\, \mathbf{M} \succeq 0\}$ per iteration, which is expensive ($\mathcal{O}(N^3)$) due to full matrix eigen-decomposition. An alternative approach eliminates the binary constraint and minimizes directly a quadratic graph smoothness term called *graph Laplacian regularization* (GLR) $\mathbf{x}^\top \mathbf{L}\mathbf{x}$ [8] for $\mathbf{x} \in \mathbb{R}^N$, and then rounds x_i's to $\{-1, 1\}$. However, in general spectral methods such as GLR do not have tight performance bounds common in SDP relaxation [9].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

$$\mathbf{M} = \begin{bmatrix} 2 & -2 & -1 \\ -2 & 5 & -2 \\ -1 & -2 & 4 \end{bmatrix}$$

$$\mathbf{SMS}^{-1} = \begin{bmatrix} 2 & -1.301 & -0.5912 \\ -3.0746 & 5 & -1.8176 \\ -1.6915 & -2.2007 & 4 \end{bmatrix}$$

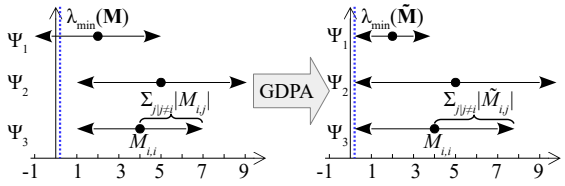

Figure 1: Example of a PD matrix $\mathbf{M}$ and its similarity transform $\tilde{\mathbf{M}} = \mathbf{SMS}^{-1}$, and their respective Gershgorin discs Ψ_i. Note that Gershgorin disc left-ends of $\tilde{\mathbf{M}}$ are aligned at $\lambda_{\min}(\mathbf{M}) = 0.1078$.

To ensure matrix variable $\mathbf{M}$ is PSD without eigen-decomposition, one naïve approach is to enforce linear constraints derived directly from the *Gershgorin circle theorem* (GCT) [10]. By GCT, every real eigenvalue λ of a real symmetric matrix $\mathbf{M}$ resides inside at least one *Gershgorin disc* Ψ_i—corresponding to row i of $\mathbf{M}$—with center $c_i(\mathbf{M}) \triangleq M_{i,i}$ and radius $r_i(\mathbf{M}) \triangleq \sum_{j \neq i} |M_{i,j}|$, *i.e.*,

$$c_i(\mathbf{M}) - r_i(\mathbf{M}) \leq \lambda \leq c_i(\mathbf{M}) + r_i(\mathbf{M}), \quad \exists i. \tag{1}$$

The corollary is that the smallest eigenvalue, $\lambda_{\min}(\mathbf{M})$, of $\mathbf{M}$ is lower-bounded by the smallest Gershgorin disc left-end, denoted by $\lambda^{-}_{\min}(\mathbf{M})$, *i.e.*,

$$\lambda^{-}_{\min}(\mathbf{M}) \triangleq \min_i c_i(\mathbf{M}) - r_i(\mathbf{M}) \leq \lambda_{\min}(\mathbf{M}). \tag{2}$$

Thus, to ensure $\mathbf{M} \succeq 0$, one can impose the sufficient condition $\lambda^{-}_{\min}(\mathbf{M}) \geq 0$. While replacing the PSD cone constraint with a set of N linear constraints, $c_i(\mathbf{M}) - r_i(\mathbf{M}) \geq 0, \forall i$, is attractive computationally, GCT lower bound $\lambda^{-}_{\min}(\mathbf{M})$ tends to be loose. As an example, consider the *positive definite* (PD) matrix $\mathbf{M}$ in Fig. 1(a) with $\lambda_{\min}(\mathbf{M}) = 0.1078$ [11]. The first Gershgorin disc left-end is $c_1(\mathbf{M}) - r_1(\mathbf{M}) = 2 - 3 = -1$, and $\lambda^{-}_{\min}(\mathbf{M}) < 0$. Thus, imposing $\lambda^{-}_{\min}(\mathbf{M}) \geq 0$ directly would unnecessarily restrict the search space and result in a sub-optimal solution to the posed problem.

A recent linear algebraic theory called *Gershgorin disc perfect alignment* (GDPA) [11] provides a theoretical foundation to tighten the GCT lower bound. Specifically, GDPA states that given a graph Laplacian matrix $\mathbf{L}$ corresponding to a balanced signed graph $\mathcal{G}$ [12], one can perform a *similarity transform*[1], $\tilde{\mathbf{L}} = \mathbf{SLS}^{-1}$, where $\mathbf{S} = \text{diag}(v_1^{-1}, \ldots, v_N^{-1})$ and $\mathbf{v}$ is the first eigenvector of $\mathbf{L}$, such that the Gershgorin disc left-ends of $\tilde{\mathbf{L}}$ are exactly aligned at $\lambda_{\min}(\mathbf{L}) = \lambda_{\min}(\tilde{\mathbf{L}})$. This means that transformed $\tilde{\mathbf{L}}$ satisfies $\lambda^{-}_{\min}(\tilde{\mathbf{L}}) = \lambda_{\min}(\tilde{\mathbf{L}})$; *i.e.*, *the GCT lower bound is the tightest possible after an appropriate similarity transform*. Continuing our example, similarity transform $\tilde{\mathbf{M}} = \mathbf{SMS}^{-1}$ of $\mathbf{M}$ has all its disc left-ends exactly aligned at $\lambda_{\min}(\mathbf{M}) = \lambda_{\min}(\tilde{\mathbf{M}}) = 0.1078$.

Leveraging GDPA, we develop a fast projection-free algorithm for semi-supervised graph classifier learning. We first observe that the optimal solution $\mathbf{M}$ of the SDP relaxation is an *adjacency* matrix to a balanced signed graph. However, GDPA requires a Laplacian matrix, which has opposite signs in the off-diagonal terms to the corresponding adjacency matrix of the same graph. Thus, we convert the problem to its SDP dual [13] and interpret the dual variable $\mathbf{H}$ instead as a Laplacian to a balanced graph sans the last graph node. To achieve graph balance, we split the last node into two and divide the original positive and negative edges among them, resulting in a revised Laplacian $\bar{\mathbf{H}}$. We repose the SDP dual problem for solution $\bar{\mathbf{H}}$, then replace the PSD cone constraint $\bar{\mathbf{H}} \succeq 0$ with linear constraints derived from GDPA. This changes the optimization to a *linear program* (LP) per iteration that is solved efficiently using fast LP solvers [14]. Finally, we extract prediction labels from our converged LP solution $\bar{\mathbf{H}}$. Experiments show that our algorithm enjoyed a $40\times$ speedup on average over the next fastest scheme while retaining comparable label prediction performance.

2 Related Work

Graph-based classification was first studied almost two decades ago [2, 3, 4]. With the advent of *graph signal processing* (GSP) [15, 16]—spectral analysis of discrete signals residing on combinatorial graphs—interest in the problem was revived [17, 18, 19]. The problem of learning a similarity graph from data has been extensively studied [20]. We focus instead on the orthogonal problem of predicting binary labels given a similarity graph and a subset of M labels.

[1] A similarity transform $\mathbf{B} = \mathbf{SAS}^{-1}$ and the original matrix $\mathbf{A}$ share the same set of eigenvalues [10].

The graph-based binary classification problem is NP-hard in general [5]. SDP—useful in approximating various NP-hard problems [13]—provides an intuitive relaxation [6]. An interior point method tailored for the slightly more general *binary quadratic problem*[2] (BQP) has complexity $\mathcal{O}(N^{3.5}\log(1/\epsilon))$, where ϵ is the tolerable error [21]. The complexity was improved to $\mathcal{O}(N^3)$ by SDCut [22, 23] via spectrahedron-based relaxation. Replacing PSD cone constraint $\mathbf{M} \succeq 0$ with a factorization $\mathbf{M} = \mathbf{X}\mathbf{X}^\top$ was proposed in [24], but resulted in a non-convex optimization for $\mathbf{X}$ that was solved locally via alternating minimization, where in each iteration a matrix inverse of worst-case complexity $\mathcal{O}(N^3)$ was required. More recent first-order methods for SDP such as [7] used ADMM [25, 26, 27], but the iterative projection onto PSD cone requires full matrix eigen-decomposition and thus expensive. In contrast, leveraging GDPA theory [11], our algorithm is entirely projection-free.

It is known in graph spectral theory [28] that balanced signed graphs have unique spectral properties [29]; for example, the *signed graph Laplacian matrix* [30] has eigenvalue 0 iff the corresponding signed graph is balanced. In contrast, extending the original GCT [10], GDPA [11] states that the Gershgorin disc left-ends of a similarity transform $\mathbf{SMS}^{-1}$ of graph Laplacian $\mathbf{M}$ to a balanced graph can be perfectly aligned at $\lambda_{\min}(\mathbf{M})$. GDPA theory was developed for *metric learning* [31] to optimize a PD matrix $\mathbf{M}$ given a convex and differentiable objective $Q(\mathbf{M})$ so that the optimal *Mahalanobis distance* $(\mathbf{f}_i - \mathbf{f}_j)^\top \mathbf{M}(\mathbf{f}_i - \mathbf{f}_j)$ for feature vectors $\mathbf{f}_i$ and $\mathbf{f}_j$ can be defined. This paper leverages GDPA [11] in an entirely different direction for graph-based binary classifier learning. Specifically, observing that solution matrix $\mathbf{H}$ to the SDP dual is a Laplacian to a balanced graph $\mathcal{G}$ sans the last graph node, we augment the last node to obtain an overall balanced graph $\bar{\mathcal{G}}$, and solve a modified SDP dual for Laplacian $\bar{\mathbf{H}}$ to $\bar{\mathcal{G}}$ via GDPA linearization.

3 Preliminaries

3.1 Graph Definitions

A graph is defined as $\mathcal{G}(\mathcal{V}, \mathcal{E})$, with node set $\mathcal{V} = \{1 \dots, N\}$, and edge set $\mathcal{E} = \{(i,j)\}$, where (i,j) means nodes i and j are connected with weight $w_{i,j} \in \mathbb{R}$. A node i may have self-loop of weights $u_i \in \mathbb{R}$. Denote by $\mathbf{W}$ the *adjacency matrix*, where $W_{i,j} = w_{i,j}$ and $W_{i,i} = u_i$. We assume that edges are undirected, and $\mathbf{W}$ is symmetric. Define next the diagonal *degree matrix* $\mathbf{D}$, where $D_{i,i} = \sum_j W_{i,j}$. The *combinatorial graph Laplacian matrix* [15] is then defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$. To account for self-loops, the *generalized graph Laplacian matrix* is defined as $\mathcal{L} = \mathbf{D} - \mathbf{W} + \text{diag}(\mathbf{W})$. Note that any real symmetric matrix can be interpreted as a generalized graph Laplacian matrix.

The *graph Laplacian regularizer* (GLR) [8] that quantifies smoothness of signal $\mathbf{x} \in \mathbb{R}^N$ w.r.t. graph specified by $\mathcal{L}$ is

$$\mathbf{x}^\top \mathcal{L}\mathbf{x} = \sum_{(i,j)\in\mathcal{E}} w_{i,j}(x_i - x_j)^2 + \sum_{i\in\mathcal{V}} u_i x_i^2. \tag{3}$$

GLR is also the objective of our graph-based classification problem.

3.2 Iterative GDPA Linearization

Denote by $\mathcal{L}$ a generalized graph Laplacian matrix to a balanced and connected signed graph $\mathcal{G}$ (with or without self-loops). A balanced graph is a graph with no cycle of odd number of negative edges. By *Cartwright-Harary Theorem* (CHT) [12], a graph is balanced iff nodes can be colored into blue and red, such that each positive (negative) edge connects nodes of the same (different) colors. GDPA [11] states that a similarity transform $\tilde{\mathcal{L}} = \mathbf{S}\mathcal{L}\mathbf{S}^{-1}$, where $\mathbf{S} = \text{diag}(v_1^{-1}, \dots, v_N^{-1})$ and $\mathbf{v}$ is the first eigenvector of $\mathcal{L}$, has its Gershgorin disc left-ends aligned exactly at $\lambda_{\min}(\mathcal{L})$, *i.e.*,

$$\tilde{\mathcal{L}}_{i,i} - \sum_{j\neq i} |\tilde{\mathcal{L}}_{i,j}| = \mathcal{L}_{i,i} - \sum_{j\neq i} |s_i \mathcal{L}_{i,j}/s_j| = \lambda_{\min}(\mathcal{L}), \quad \forall i \in \{1, \dots, N\}. \tag{4}$$

To solve an optimization of the form $\min_{\mathcal{L}\succeq 0} Q(\mathcal{L})$, one can leverage GDPA and optimize iteratively as follows. At iteration t with solution $\mathcal{L}^t$, compute first eigenvector $\mathbf{v}^t$ to $\mathcal{L}^t$ corresponding to $\lambda_{\min}(\mathcal{L}^t)$; extreme eigenvector $\mathbf{v}^t$ can be efficiently computed in complexity $\mathcal{O}(ab)$ using *Locally*

[2] BQP objective takes a quadratic form $\mathbf{x}^\top \mathbf{Q}\mathbf{x}$, but $\mathbf{Q}$ is not required to be a Laplacian to a similarity graph.

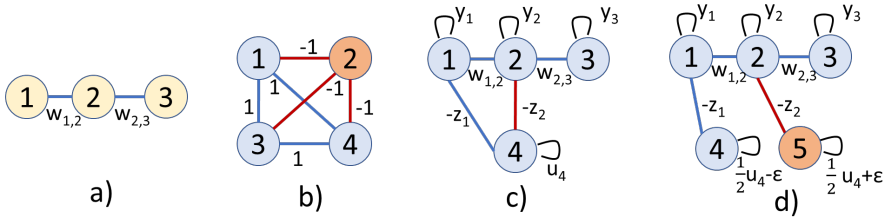

Figure 2: (a) 3-node line graph example. (b) Ideal solution $\mathbf{M}$ to SDP primal (8) as adjacency matrix. (c) Solution $\mathbf{H}$ to SDP dual (12) as Laplacian matrix. (d) Solution $\bar{\mathbf{H}}$ to modified SDP dual (20) as Laplacian matrix. Positive / negative edges are colored in blue / red. Self-loop weight u_4 in (c) for node 4 is $u_4 = y_4 + z_1 + z_2$.

Optimal Block Preconditioned Conjugate Gradient (LOBPCG) [32], where a is the number of non-zero entries in $\mathcal{L}^t$ and b is the iteration number till convergence[3]. Define scalars $s_i^t = 1/v_i^t, \forall i$. Then for iteration $t+1$, solve the following optimization:

$$\min_{\mathcal{L}} Q(\mathcal{L}), \qquad \text{s.t. } \mathcal{L}_{i,i} - \sum_{j \neq i} |s_i^t \mathcal{L}_{i,j} / s_j^t| \geq 0, \quad \forall i \in \{1, \ldots, N\}. \tag{5}$$

Linear constraints in (5) ensure that the similarity transform $\tilde{\mathcal{L}} = \mathbf{S}\mathcal{L}\mathbf{S}^{-1}$ is PSD by GCT, and hence solution $\mathcal{L}$ is PSD. Since scalars $\{s_i^t\}$ are computed from first eigenvector $\mathbf{v}^t$ of $\mathcal{L}^t \succeq 0$, by GDPA $\mathbf{S}\mathcal{L}^t\mathbf{S}^{-1}$ has all its disc left-ends aligned exactly at $\lambda_{\min}(\mathcal{L}^t) \geq 0$, and hence $\mathcal{L}^t$ remains feasible at iteration $t+1$. Thus, objective $Q(\mathcal{L}^t)$ is monotonically non-increasing with t, and the algorithm converges to a local minimum. We invoke this iteration to solve our posed SDP dual as well.

4 Formulation of Graph-based Classifier Learning

We first formulate the graph-based classifier learning problem and relax it to an SDP problem in Section 4.1. We then present its SDP dual with dual variable matrix $\mathbf{H}$ in Section 4.2. Finally, we interpret $\mathbf{H}$ as a graph Laplacian, and augment its corresponding graph $\mathcal{G}$ to a balanced graph $\bar{\mathcal{G}}$ for GDPA linearization in Section 4.3.

4.1 SDP Primal

Given a PSD graph Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N \times N}$ of a positive similarity graph $\mathcal{G}^o$ (*i.e.*, all edge weights $w_{i,j} \geq 0$), one can formulate a graph-based binary classification problem as follows:

$$\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{L} \mathbf{x}, \quad \text{s.t.} \begin{cases} x_i^2 = 1, \ \forall i \in \{1, \ldots, N\} \\ x_i = \hat{x}_i, \ \forall i \in \{1, \ldots, M\} \end{cases}. \tag{6}$$

where $\{\hat{x}_i\}_{i=1}^M$ are the M known labels. The objective in (6) dictates that signal $\mathbf{x}$ is smooth w.r.t. graph $\mathcal{G}^o$ specified by $\mathbf{L}$. Because $\mathbf{L}$ is PSD [16], the objective is lower-bounded by 0, *i.e.*, $\mathbf{x}^\top \mathbf{L} \mathbf{x} \geq 0, \forall \mathbf{x} \in \mathbb{R}^N$. The first binary constraint ensures $x_i \in \{-1, 1\}$. The second constraint ensures that entries x_i in signal $\mathbf{x}$ agrees with known labels $\{\hat{x}_i\}_{i=1}^M$.

As an example, consider a 3-node line graph shown in Fig. 2(a), where edges $(1,2)$ and $(2,3)$ have weights $w_{1,2}$ and $w_{2,3}$, respectively. The adjacency matrix $\mathbf{W}$ and graph Laplacian matrix $\mathbf{L}$ are:

$$\mathbf{W} = \begin{bmatrix} 0 & w_{1,2} & 0 \\ w_{1,2} & 0 & w_{2,3} \\ 0 & w_{2,3} & 0 \end{bmatrix}, \qquad \mathbf{L} = \begin{bmatrix} d_1 & -w_{1,2} & 0 \\ -w_{1,2} & d_2 & -w_{2,3} \\ 0 & -w_{2,3} & d_3 \end{bmatrix} \tag{7}$$

where $d_i = \sum_{j|(i,j) \in \mathcal{E}} w_{i,j}$ is the degree of node i. Suppose known labels are $\hat{x}_1 = 1$ and $\hat{x}_2 = -1$.

Due to the binary constraint on x_i's, (6) is NP-hard [5]. One can define an SDP relaxation [5] as follows. Define first $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$ and $\mathbf{M} = [\mathbf{X} \ \ \mathbf{x}; \ \mathbf{x}^\top \ \ 1]$. $\mathbf{M}$ is PSD because: i) block [1] is PSD, and ii) the *Schur complement* of block [1] of $\mathbf{M}$ is $\mathbf{X} - \mathbf{x}\mathbf{x}^\top = \mathbf{0}$, which is also PSD. Thus, the two

[3] *Warm start* [11] can be employed to reduce b in subsequent iterations given $\mathbf{v}^t$ is computed repeatedly for gradually changing $\mathcal{L}^t$'s. See Section 5 for details.

constraints $\mathbf{M} \succeq 0$ and $\text{rank}(\mathbf{X}) = 1$ is equivalent to $\mathbf{X} = \mathbf{x}\mathbf{x}^\top$, which together with $X_{ii} = 1, \forall i$ implies $x_i^2 = 1, \forall i$. To convexify the problem, we drop the non-convex rank constraint and write the SDP relaxation for optimization variable $\mathbf{M}$ as

$$\min_{\mathbf{x},\mathbf{X}} \text{Tr}(\mathbf{L}\mathbf{X}) \;\; \text{s.t.} \;\; \begin{cases} X_{ii} = 1, i \in \{1, \dots, N\} \\ \mathbf{M} \triangleq \begin{bmatrix} \mathbf{X} & \mathbf{x} \\ \mathbf{x}^\top & 1 \end{bmatrix} \succeq 0 \\ x_i = \hat{x}_i, \;\; i \in \{1, \dots, M\} \end{cases} \tag{8}$$

where $\text{Tr}(\mathbf{x}^\top \mathbf{L}\mathbf{x}) = \text{Tr}(\mathbf{L}\mathbf{x}\mathbf{x}^\top) = \text{Tr}(\mathbf{L}\mathbf{X})$. Because (8) has linear objective and constraints with an additional PSD cone constraint, $\mathbf{M} \succeq 0$, it is an SDP problem. We call (8) the SDP primal.

Continuing our example, consider ground-truth labels $\mathbf{x} = [1 \;\; -1 \;\; 1]^\top$ for the 3-node graph in Fig. 2(a). The corresponding solution matrix $\mathbf{M} = [\mathbf{x}\mathbf{x}^\top \; \mathbf{x}; \; \mathbf{x}^\top \; 1]$ is

$$\mathbf{M} = \begin{bmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix}. \tag{9}$$

Observe that $\mathbf{M}$ can be interpreted as an *adjacency* matrix to a balanced signed graph; nodes 1, 3 and 4 can be colored blue, and node 2 can be colored red, so that positive (negative) edges connect only nodes of the same (different) colors. See Fig. 2(b) for an illustration of the corresponding signed graph when interpreting $\mathbf{M}$ as an adjacency matrix (self-loops are not shown). However, while the solution space for the SDP primal (8) exhibits desirable graph balance, GDPA requires instead a graph Laplacian matrix to a balanced graph, which has opposite signs in the off-diagonal terms as the adjacency matrix. This motivates us to investigate the corresponding SDP dual problem instead.

4.2 SDP Dual

We derive the dual problem based on SDP duality theory [13]. We first define

$$\mathbf{A}_i = \text{diag}(\mathbf{e}_{N+1}(i)), \quad \mathbf{B}_i = \begin{bmatrix} \mathbf{0}_{N\times N} & \mathbf{e}_N(i) \\ \mathbf{e}_N^\top(i) & 0 \end{bmatrix}. \tag{10}$$

where $\mathbf{e}_N(i) \in \{0,1\}^N$ is a length-N binary *canonical vector* with a single non-zero entry equals to 1 at the i-th entry, $\mathbf{0}_{N\times N}$ is a N-by-N matrix of zeros, and $\text{diag}(\mathbf{v})$ is a diagonal matrix with diagonal entries equal to $\mathbf{v}$. Note that $\mathbf{A}_i$ and $\mathbf{B}_i$ are symmetric. Next, we collect M known labels $\{\hat{x}_i\}_{i=1}^M$ into a vector $\mathbf{b} \in \mathbb{R}^M$ of length M, *i.e.*,

$$b_i = 2\hat{x}_i, \quad \forall i \in \{1, \dots, M\}. \tag{11}$$

We now define the SDP dual of (8) as

$$\min_{\mathbf{y},\mathbf{z}} \;\; \mathbf{1}_{N+1}^\top \mathbf{y} + \mathbf{b}^\top \mathbf{z}, \qquad \text{s.t.} \;\; \mathbf{H} \triangleq \sum_{i=1}^{N+1} y_i \mathbf{A}_i + \sum_{i=1}^{M} z_i \mathbf{B}_i - \mathbf{L} \succeq 0 \tag{12}$$

where $\mathbf{1}_N$ is a length-N vector of ones, and dual variables are $\mathbf{y} \in \mathbb{R}^{N+1}$ and $\mathbf{z} \in \mathbb{R}^M$. Because the objective is a minimization, when $b_i < 0$ (*i.e.*, $\hat{x}_i < 0$), the corresponding $z_i \geq 0$. Similarly, for $b_i > 0$, $z_i \leq 0$. Thus, the signs of variables z_i's are known *a priori*. Without loss of generality, we assume $z_i \leq 0, \forall i \in \{1, \dots, M_1\}$ and $z_i \geq 0, \forall i \in \{M_1 + 1, \dots, M\}$ in the sequel.

4.3 Reformulating the SDP Dual

We interpret $\mathbf{H} \in \mathbb{R}^{(N+1)\times(N+1)}$ in (12) as a graph Laplacian corresponding to a graph $\mathcal{G}$. However, $\mathcal{G}$ is not a balanced signed graph, because of the last row / column in $\mathbf{H}$. To see this, we write

$$\mathbf{H} = \begin{bmatrix} \mathcal{L}_y & \mathbf{g} \\ \mathbf{g}^\top & y_{N+1} \end{bmatrix} \tag{13}$$

where $\mathbf{g} = [z_1 \dots z_M \; \mathbf{0}_{N-M}^\top]^\top$. Matrix $\mathcal{L}_y \in \mathbb{R}^{N\times N}$, which equals to $\mathcal{L}_y = \text{diag}(y_1, \dots, y_N) + \mathbf{L}$, is a generalized Laplacian to a N-node positive graph $\mathcal{G}^+$. However, node $N+1$ has *both* positive

and negative edges to $\mathcal{G}^+$ stemming from negative z_i's and positive z_i's, respectively. As a result, $\mathbf{H}$ is not a Laplacian corresponding to a balanced signed graph.

Continuing our 3-node line graph example with Laplacian $\mathbf{L}$, the corresponding $\mathcal{L}_y$ and $\mathbf{H}$ are

$$\mathcal{L}_y = \begin{bmatrix} y_1+d_1 & -w_{1,2} & 0 \\ -w_{1,2} & y_2+d_2 & -w_{2,3} \\ 0 & -w_{2,3} & y_3+d_3 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} y_1+d_1 & -w_{1,2} & 0 & z_1 \\ -w_{1,2} & y_2+d_2 & -w_{2,3} & z_2 \\ 0 & -w_{2,3} & y_3+d_3 & 0 \\ z_1 & z_2 & 0 & y_4 \end{bmatrix}. \quad (14)$$

Interpreting $\mathbf{H}$ as a graph Laplacian, node 4 has degree $d_4 = -z_1 - z_2$. Thus, $y_4 = u_4 + d_4$, and self-loop weight for node 4 iss $u_4 = y_4 + z_1 + z_2$. See Fig. 2(c) for an illustration of this graph $\mathcal{G}$.

In graph terminology, node $(N+1)$ has positive and negative edges, with respective weights $\{-z_i\}_{i=1}^{M_1}$ and $\{-z_i\}_{i=M_1+1}^{M}$, to $\mathcal{G}^+$, and a self-loop with weight $u_{N+1} = y_{N+1} + \sum_{i=1}^{M} z_i$. We construct an augmented graph $\bar{\mathcal{G}}$ with $N+2$ nodes from $\mathcal{G}$ by splitting node $N+1$ in $\mathcal{G}$ into two in $\bar{\mathcal{G}}$, dividing positive and negative edges between them. The specific graph construction for $\bar{\mathcal{G}}$ procedure is

1. Construct first N nodes with the same inter-connections as sub-graph $\mathcal{G}^+$.
2. Construct node $N+1$ with positive edges $\{-z_i\}_{i=1}^{M_1}$ and node $N+2$ with negative edges $\{-z_i\}_{i=M_1+1}^{M}$ to the first N nodes in sub-graph $\mathcal{G}^+$.
3. Add self-loops for node $N+1$ and $N+2$ with respective weights $u_{N+1}/2-\epsilon$ and $u_{N+1}/2+\epsilon$, where $\epsilon \in \mathbb{R}$ is a parameter to be discussed.

Denote by $\bar{\mathbf{H}} \in \mathbb{R}^{(N+2)\times(N+2)}$ the graph Laplacian matrix corresponding to $\bar{\mathcal{G}}$. Continuing our 3-node graph example, Fig. 2(d) shows the augmented graph $\bar{\mathcal{G}}$, and the corresponding $\bar{\mathbf{H}}$ is

$$\bar{\mathbf{H}} = \begin{bmatrix} y_1+d_1 & -w_{1,2} & 0 & z_1 & 0 \\ -w_{1,2} & y_2+d_2 & -w_{2,3} & 0 & z_2 \\ 0 & -w_{2,3} & y_3+d_3 & 0 & 0 \\ z_1 & 0 & 0 & \frac{1}{2}(y_4 - z_1 + z_2) - \epsilon & 0 \\ 0 & z_2 & 0 & 0 & \frac{1}{2}(y_4 + z_1 - z_2) + \epsilon \end{bmatrix}. \quad (15)$$

Spectrally, $\bar{\mathbf{H}}$ and $\mathbf{H}$ are related; we prove that $\lambda_{\min}(\bar{\mathbf{H}})$ is a lower bound for $\lambda_{\min}(\mathbf{H})$.

Lemma 1. *The smallest eigenvalue $\lambda_{\min}(\bar{\mathbf{H}})$ of graph Laplacian $\bar{\mathbf{H}}$ to augmented graph $\bar{\mathcal{G}}$ is a lower bound for $\lambda_{\min}(\mathbf{H})$ of Laplacian $\mathbf{H}$ to $\mathcal{G}$, i.e.,*

$$\lambda_{\min}(\bar{\mathbf{H}}) \leq \lambda_{\min}(\mathbf{H}). \quad (16)$$

Proof. Denote by $\mathcal{G}$ the graph represented by generalized graph Laplacian $\mathbf{H}$, with inter-node edge weights $\{w_{ij}\}$ and self-loop weights $\{u_i\}$. Denote by $\mathbf{v} \in \mathbb{R}^{N+1}$ the first eigenvector of $\mathbf{H}$ corresponding to the smallest eigenvalue $\lambda_{\min}(\mathbf{H})$. From (3), GLR of $\mathbf{H}$ computed using $\mathbf{v}$ is

$$\mathbf{v}^\top \mathbf{H} \mathbf{v} = \sum_{(i,j)\in\mathcal{E} \mid 1\leq i,j\leq N} w_{i,j}(v_i - v_j)^2 - \sum_{i=1}^{M} z_i(v_{N+1} - v_i)^2 + \sum_{i=1}^{N} y_i v_i^2 + u_{N+1} v_{N+1}^2. \quad (17)$$

Now construct $\boldsymbol{\alpha} \in \mathbb{R}^{N+2}$, where $\boldsymbol{\alpha} = [v_1 \dots v_N \; v_{N+1} \; v_{N+1}]^\top$. GLR of $\bar{\mathbf{H}}$ computed using $\boldsymbol{\alpha}$ is

$$\begin{aligned} \boldsymbol{\alpha}^\top \bar{\mathbf{H}} \boldsymbol{\alpha} = & \sum_{(i,j)\in\mathcal{E} \mid 1\leq i,j\leq N} w_{i,j}(v_i - v_j)^2 - \sum_{i=1}^{M_1} z_i(v_{N+1} - v_i)^2 - \sum_{i=M_1+1}^{M} z_i(v_{N+1} - v_i)^2 \\ & + \sum_{i=1}^{N} y_i v_i^2 + \left(\frac{u_{N+1}}{2} - \epsilon\right) v_{N+1}^2 + \left(\frac{u_{N+1}}{2} + \epsilon\right) v_{N+1}^2. \end{aligned} \quad (18)$$

Thus, $\mathbf{v}^\top \mathbf{H} \mathbf{v} = \boldsymbol{\alpha}^\top \bar{\mathbf{H}} \boldsymbol{\alpha}$. Since first eigenvector $\mathbf{v}$ minimizes the Rayleigh quotient of $\mathbf{H}$,

$$\lambda_{\min}(\mathbf{H}) = \frac{\mathbf{v}^\top \mathbf{H} \mathbf{v}}{\mathbf{v}^\top \mathbf{v}} \overset{(a)}{\geq} \frac{\boldsymbol{\alpha}^\top \bar{\mathbf{H}} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top \boldsymbol{\alpha}} \overset{(b)}{\geq} \lambda_{\min}(\bar{\mathbf{H}}). \quad (19)$$

(a) holds since $\mathbf{v}^\top \mathbf{v} \leq \boldsymbol{\alpha}^\top \boldsymbol{\alpha}$ by construction, and (b) holds since $\lambda_{\min}(\bar{\mathbf{H}}) = \min_{\mathbf{x}} \frac{\mathbf{x}^\top \bar{\mathbf{H}} \mathbf{x}}{\mathbf{x}^\top \mathbf{x}}$. □

From the proof above, the usefulness of parameter ϵ becomes clear: the bound $\lambda_{\min}(\bar{\mathbf{H}}) \leq \lambda_{\min}(\mathbf{H})$ becomes tight when the last two entries in the first eigenvector of $\bar{\mathbf{H}}$ are similar. To promote this, we set ϵ to an appropriate large value, so that the first eigenvector minimizing the Rayleigh quotient of $\bar{\mathbf{H}}$ would choose similar small values for the last two entries.

Given Lemma 1, we now reformulate the SDP dual (12) by keeping the same objective but imposing PSD cone constraint on $\bar{\mathbf{H}}$ instead of $\mathbf{H}$. Define $\mathbf{A}'_i$, $\mathbf{B}'_i$ and $\mathbf{B}''_i$ similarly to (10) but for a larger $(N+2)$-by-$(N+2)$ matrix; *i.e.*, $\mathbf{A}'_i = \text{diag}(\mathbf{e}_{N+2}(i))$, $\mathbf{B}'_i = [\mathbf{B}_i \ \mathbf{0}_{N+1}; \mathbf{0}^\top_{N+1} \ 0]$, and $\mathbf{B}''_i = [\mathbf{0}_{(N+1)\times(N+1)} \ \mathbf{e}_{N+1}(i); \mathbf{e}^\top_{N+1}(i) \ 0]$. The reformulated SDP dual is

$$\begin{aligned} &\min_{\mathbf{y},\mathbf{z}} \ \mathbf{1}^\top_{N+1}\mathbf{y} + \mathbf{b}^\top\mathbf{z}, \\ &\text{s.t. } \bar{\mathbf{H}} \triangleq \sum_{i=1}^{N} y_i\mathbf{A}'_i + \kappa_{N+1}\mathbf{A}'_{N+1} + \kappa_{N+2}\mathbf{A}'_{N+2} + \sum_{i=1}^{M_1} z_i\mathbf{B}'_i + \sum_{i=M_1+1}^{M} z_i\mathbf{B}''_i - \mathbf{L} \succeq 0 \end{aligned} \tag{20}$$

where $\kappa_{N+1} = \frac{u_{N+1}}{2} - \sum_{i=1}^{M_1} z_i - \epsilon$ and $\kappa_{N+2} = \frac{u_{N+1}}{2} - \sum_{i=M_1+1}^{M} z_i + \epsilon$. Given $\bar{\mathbf{H}}$ is now a Laplacian to a balanced graph, we discuss the application of GDPA linearization to solve (20) next.

5 Algorithm Implementation

5.1 GDPA Linearization

We replace the PSD cone constraint on $\bar{\mathbf{H}}$ in (20) with $N+2$ linear constraints via GDPA [11]. Specifically, at iteration t, we compute first eigenvector $\mathbf{v}^t$ of solution $\bar{\mathbf{H}}^t$ using LOBPCG [32]. We define scalars $s_i = 1/v^t_i, \forall i \in \{1,\dots,N+2\}$. Finally, we write $N+2$ constraints corresponding to $\lambda^-_{\min}(\mathbf{S}\bar{\mathbf{H}}\mathbf{S}^{-1}) \geq 0$, where $\mathbf{S} = \text{diag}(s_1,\dots,s_{N+2})$, *i.e.*,

$$\begin{aligned} y_i + d_i - \textstyle\sum_{j\neq i} |s_i w_{i,j}/s_j| - |s_i z_i/s_{N+1}| &\geq 0, \quad \forall i \in \{1,\dots,M_1\} \\ y_i + d_i - \textstyle\sum_{j\neq i} |s_i w_{i,j}/s_j| - |s_i z_i/s_{N+2}| &\geq 0, \quad \forall i \in \{M_1+1,\dots,M\} \\ y_i + d_i - \textstyle\sum_{j\neq i} |s_i w_{i,j}/s_j| &\geq 0, \quad \forall i \in \{M+1,\dots,N\} \\ u_{N+1}/2 - \epsilon - \textstyle\sum_{j=1}^{M_1} |s_{N+1} z_j/s_j| &\geq 0 \\ u_{N+1}/2 + \epsilon - \textstyle\sum_{j=M_1+1}^{M} |s_{N+2} z_j/s_j| &\geq 0 \end{aligned} \tag{21}$$

where the indices for summation $\sum_{j\neq i}$ are $\{1,\dots,N\} \setminus i$. Note that the absolute value operation can be appropriately removed for each term $s_i w_{i,j}/s_j$ and $s_i z_i/s_j$, since the signs for s_i, $w_{i,j}$ and z_i are known. Together with linear objective in (20), this constitutes an LP for variables $\mathbf{y}$ and $\mathbf{z}$, solvable using any available fast LP solvers [14]. Compared to SDP primal (8) with a large matrix variable $\mathbf{M} \in \mathbb{R}^{(N+1)\times(N+1)}$, our LP variables, $\mathbf{y} \in \mathbb{R}^{N+1}$ and $\mathbf{z} \in \mathbb{R}^M$, are much smaller.

A sequence of LPs are solved, each time with scalars s_i's updated from computed solution $\bar{\mathbf{H}}^t$, until convergence. The bulk of the complexity resides in the computation of the first eigenvector $\mathbf{v}^t$ for each LP solution $\bar{\mathbf{H}}^t$. LOBPCG is an iterative algorithm that can benefit from *warm start* [11]: with a good initial guess for $\mathbf{v}^t$, the algorithm converges faster. Since $\bar{\mathbf{H}}^t$ changes gradually through our iterations, we use previously computed eigenvector $\mathbf{v}^{t-1}$ of $\bar{\mathbf{H}}^{t-1}$ as initial guess for $\mathbf{v}^t$ of $\bar{\mathbf{H}}^t$. Experiments show that warm start reduces the iteration number till convergence significantly.

5.2 Initialization & Prediction Label Extraction

Our LP in Section 5.1 requires an initial $\bar{\mathbf{H}}^0$ to compute first eigenvector $\mathbf{v}^0$, so that scalars $\{s_i\}_{i=1}^{N+2}$ can be defined for $N+2$ linear constraints in (21). To initialize $\bar{\mathbf{H}}^0$, we set $\mathbf{y}^0 = [\mathbf{1}^\top_M \ \mathbf{0}^\top_{N-M} \ M]^\top$ and $\mathbf{z}^0 = [-\hat{x}_1 \dots -\hat{x}_M]$. Parameter ϵ is set to $\epsilon^t = \mathbf{1}^\top_{N+1}\mathbf{y}^{t-1} + \mathbf{1}^\top_M\mathbf{z}^{t-1}$ at iteration t. $\bar{\mathbf{H}}^0$ can then be computed using definition of $\bar{\mathbf{H}}$ in (20).

As similarly done in [5], we extract prediction labels $\mathbf{x}^* = [x_1 \dots x_N]^\top$ from converged LP solution $\mathbf{y}^*$ and $\mathbf{z}^*$ as follows. We first construct $\mathbf{H}^*$ using $\mathbf{y}^*$ and $\mathbf{z}^*$ using definition of $\mathbf{H}$ in (12). We then compute $\mathbf{x}^* = \text{sign}(\hat{x}_1 v_1 \mathbf{v})$, where v_1 is the first entry of the first eigenvector $\mathbf{v}$ of $\mathbf{H}^*$. See [5] for details of recovering SDP primal variables from dual variables in BQP.

6 Experiments

6.1 Experimental Setup

We implemented our GDPA-graph-based classifier learning scheme in Matlab[4], and evaluated it in terms of average classification error rate and running time. We compared our algorithm against the following schemes that solve the SDP primal problem (8) directly: i) two primal-dual interior-point solvers for SDP, SeDuMi and MOSEK, both of which are available in CVX with a CVX Professional license [33], ii) an ADMM first-order operator-splitting solver CDCS [26, 27] with an LGPL-3.0 License [34], iii) a spectrahedron-based relaxation solver SDCut [22, 23, 35] that involves L-BFGS-B [36], and iv) a biconvex relaxation solver BCR [24, 37], all of which are implemented in Matlab. In addition, we employed CDCS again to solve our modified SDP dual problem (20).

We set the convergence threshold of the first eigenvector solver LOBPCG to be 10^{-4}, with maximum number of iterations 200. We set the convergence threshold of our LP solver to be 10^{-4} also, with maximum number of iterations 100, since first-order methods, *i.e.*, CDCS and SDCut, aim at computing a solution of moderate accuracy [26]. Accordingly, we set the convergence threshold of SeDuMi and MOSEK to be 'low', which is approximately equal to 10^{-4} and the lowest precision setting in CVX. We set the convergence thresholds of CDCS and SDCut to be 10^{-3}, the maximum number of ADMM iterations in CDCS to be 1000, the maximum number of iterations for L-BFGS-B in SDCut and the main loop in BCR to be 100, and the Frobenius norm weight in SDCut to be 100. We chose these settings since smaller convergence thresholds and larger number of iterations would cause CDCS, SDCut and BCR to be significantly slower to converge. We used default settings for all remaining solvers. All computations were carried out on a Windows 10 64bit PC with AMD RyzenThreadripper 3960X 24-core processor 3.80 GHz and 128GB of RAM.

We adopted 17 binary datasets that are freely available in UCI [38] and LibSVM [39]. For experimental efficiency, we first performed a K-fold ($K \leq 5$) split for each dataset with random seed 0, and then created 10 instances of 50% training-50% test split for each fold, with random seeds 1-10 [40]. The above setup resulted in problem sizes from 29 to 400. We applied the following two data normalization schemes for the training/test data: i) a *standardization* scheme in [41] that first subtracts the mean and divides by the feature-wise standard deviation, and then normalizes to unit length sample-wise, and ii) a *min-max* scheme [40] that rescales each feature to within 0 and 1. We added 10^{-12} noise to the dataset to avoid NaN's due to data normalization on small samples.

6.2 Experimental Results

Fig. 3 and the first two plots of Fig. 4 show classification error rates and runtime (in log scale) using min-max and standardization data re-scaling strategies for 17 different datasets, respectively. The x-axis of each plot denotes the datasets in ascending order of problem sizes. Each point in the plots denotes the average of $10K$ runs. Fig. 4 (right) shows runtime versus problem size (4 to 24428) using the same dataset `cod-rna` (freely avaiable in LibSVM [39]). We did not execute SeDuMi, MOSEK, CDCS (8), BCR, SDcut, or CDCS (20) when the problem size was larger than 976.

In terms of classification error rate for min-max re-scaling, MOSEK, CDCS (8) and SeDuMi had slightly larger error rates: 32.52%, 32.38% and 29.92%, respectively. GDPA had 29.11%, which was very close to CDCS (20) at 29.24% and SDCut at 28.76%. This shows that our proposed GDPA linearization (21) closely approximated the modified SDP dual (20) in performance. BCR at 26.82% was roughly 2% smaller. In the standardization re-scaling case, CDCS (8), MOSEK, and SeDuMi had the largest error rates: 32.75%, 32.5% and 31.0%, respectively. GDPA had 26.88%, close to CDCS (20) at 26.9% and SDCut with 26.82%. BCR at 24.8% was again roughly 2% smaller. By factorizing a PSD matrix $\mathbf{M} = \mathbf{X}\mathbf{X}^\top$, BCR avoided any SDP relaxation, which may explain its slightly better performance here. However, BCR solved a non-convex optimization problem converging to a local minimum, and thus occasionally the performance was quite poor (*e.g.*, see `sonar` in Fig. 3(left)). Overall, all solvers performed similarly given constructed similarity graphs in the two cases.

In terms of runtime, BCR was competitive with GDPA when the problem size was small, but GDPA significantly outperformed all competing solvers when the problem size was large. Specifically, the speed gain increased as problem size increased; for `madelon` with problem size 400, the speedup of

[4] results reproducible via code in `https://anonymous.4open.science/r/GDPA_matlab-EF4D/`

GDPA over the next fastest scheme BCR was $346\times$. Fig. 4 (right) also shows that the computation time for GDPA increased gracefully as the problem size increased to very large sizes. The reason for our dramatic speed gain is the fast computation of first eigenvectors using LOBPCG, which benefited from warm start during the LP iterations. In general, GDPA performed fewer than 10 LP's until convergence. In contrast, both CDCS and SDCut required eigen-decomposition of a matrix of size $N \times N$ per iteration. Because $\mathbf{L}$ described a dense graph in our experiments, the speedup of replacing the full eigen-decomposition with simpler first eigenvector computation per iteration was significant. For BCR, each iteration required either N-dimensional matrix inversion for a least-squares problem or iterative gradient descent, which was computationally expensive as the problem size increased. On average, GDPA enjoyed a $40.9\times$ speedup over the next fastest solver BCR.

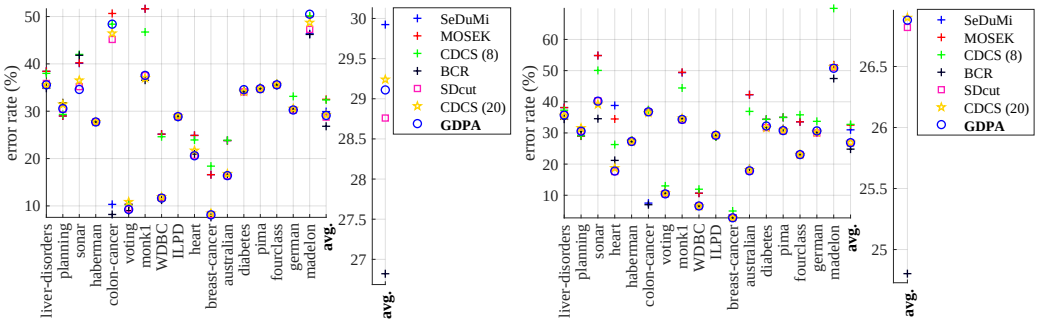

Figure 3: Error rates (%) for min-max (left) and standardization (right) data re-scaling.

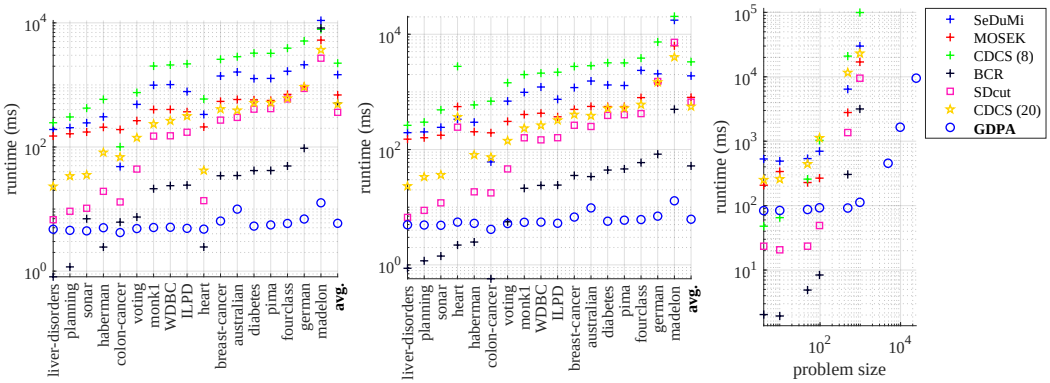

Figure 4: Runtime (ms) for min-max (left) and standardization (center) data re-scaling on different datasets, and runtime (ms) for variable problem sizes on the same dataset `cod-rna` (right).

7 Conclusion

We propose a fast projection-free algorithm for the graph-based classifier learning problem. The key idea is to replace the difficult-to-compute positive semi-definite (PSD) cone constraint with linear constraints derived from the recent Gershgorin disc perfect alignment (GDPA) theory, so that the optimization can be solved as a sequence of linear programs (LP). Experiments show that our algorithm enjoyed a considerable speedup while retaining comparable label prediction performance.

A graph classifier scalable to very large sizes encourages ubiquitous deployment for wide-ranging applications. Negative social impact can result if the tool is misused by enabling classification for discriminatory purposes. As an optimization problem, graph-based binary classification is rather narrowly defined (though multi-class classification can be implemented as a tree of binary classifiers). Furthermore, good performance depends heavily on the construction of a good similarity graph, which is outside the scope of this paper. However, we conjecture that the general methodology of GDPA linearization can be similarly tailored to other SDP problems with PSD cone constraints. We anticipate that speedups in other SDP problems will also be significant.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Berlin, Heidelberg, 2006.

[2] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf, "Learning with local and global consistency," in *16th International Conference on Neural Information Processing (NIPS)*, Whistler, Canada, December 2003.

[3] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semisupervised learning on large graphs," in *Shawe-Taylor J., Singer Y. (eds) Learning Theory, COLT 2004, Lecture Notes in Computer Science*, 2004, vol. 3120, pp. 624–638.

[4] A. Guillory and J. Bilmes, "Label selection on graphs," in *Twenty-Third Annual Conference on Neural Information Processing Systems*, Vancouver, Canada, December 2009.

[5] Z. Luo, W. Ma, A. M. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 20–34, 2010.

[6] Z. Li, J. Liu, and X. Tang, "Pairwise constraint propagation by semidefinite programming for semi-supervised classification," in *ACM International Conferene on Machine Learning*, Helsinki, Finland, July 2008.

[7] B O'Donoghue, E. Chu, N. Parikh nad, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," in *Journal of Optimization Theory and Applications*, 2016, vol. 169, no.3, pp. 1042–1068.

[8] J. Pang and G. Cheung, "Graph Laplacian regularization for inverse imaging: Analysis in the continuous domain," in *IEEE Transactions on Image Processing*, April 2017, vol. 26, no.4, pp. 1770–1785.

[9] M. Goemans and D. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, pp. 1115–1145, Nov. 1995.

[10] R. S. Varga, *Gershgorin and his circles*, Springer, 2004.

[11] C. Yang, G. Cheung, and H. Wei, "Signed graph metric learning via Gershgorin disc perfect alignment," *arXiv*, 2021.

[12] D. Cartwright and F. Harary, "Structural balance: a generalization of Heider's theory," in *Psychological Review*, 1956, vol. 63, no.5, pp. 277–293.

[13] B. Gartner and J. Matousek, *Approximation Algorithms and Semidefinite Programming*, Springer, 2012.

[14] R. Vanderbei, *Linear Programming: Foundations and Extensions (5th Edition)*, Springer Nature, 2021.

[15] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," in *Proceedings of the IEEE*, May 2018, vol. 106, no.5, pp. 808–828.

[16] G. Cheung, E. Magli, Y. Tanaka, and M. Ng, "Graph spectral image processing," in *Proceedings of the IEEE*, May 2018, vol. 106, no.5, pp. 907–930.

[17] M. Gavish, B. Nadler, and R. Coifman, "Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi-supervised learning," in *27th International Conference on Machine Learning*, Haifa, Israel, June 2010.

[18] D. Shuman, M. Faraji, and P. Vandergheynst, "Semi-supervised learning with spectral graph wavelets," in *International Conference on Sampling Theory and Applications (SampTA)*, Singapore, May 2011.

[19] G. Cheung, W.-T. Su, Y. Mao, and C.-W. Lin, "Robust semisupervised graph classifier learning with negative edge weights," in *IEEE Transactions on Signal and Information Processing over Networks*, December 2018, vol. 4, no.4, pp. 712–726.

[20] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 44–63, 2019.

[21] C. Helmberg, F. Rendl, R. Vanderbei, and H. Wolkowicz, "An interior-point method for semidefinite programming," in *SAIM J. Optim.*, 1996, vol. 6, no.2, pp. 342–361.

[22] P. Wang, C. Shen, and A. van den Hengel, "A fast semidefinite approach to solving binary quadratic problems," in *IEEE International Conference on Computer Vision and Pattern Recognition*, Portland, OR, June 2013.

[23] P. Wang, C. Shen, A. Hengel, and P. Torr, "Large-scale binary quadratic optimization using semidefinite relaxation and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 3, pp. 470–485, 2017.

[24] S. Shah et al., "Biconvex relaxation for semidefinite programming in computer vision," in *European Conference on Computer Vision*, Amsterdam, the Netherlands, October 2016.

[25] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," in *Foundations and Trends in Optimization*, 2011, vol. 3, no.1, pp. 1–122.

[26] Y. Zheng, G. Fantuzzi, and A. Papachristodoulou, "Fast ADMM for sum-of-squares programs using partial orthogonality," *IEEE Transactions on Automatic Control*, vol. 64, no. 9, pp. 3869–3876, 2019.

[27] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart, and A. Wynn, "Chordal decomposition in operator-splitting methods for sparse semidefinite programs," *Mathematical Programming*, vol. 180, pp. 489—-532, 2020.

[28] F. Chung, *Spectral Graph Theory*, American Mathematical Society, 1996.

[29] T. Dittrich and G. Matz, "Signal processing on signed graphs: Fundamentals and potentials," in *IEEE Signal Processing Magazine*, November 2020, vol. 37, no.6, pp. 86–98.

[30] J. Kunegis, S. Schmidt, A. Lommatzsch, J. Lerner, E. D. Luca, and S. Albayrak, "Spectral analysis of signed graphs for clustering, prediction and visualization," in *SIAM International Conference on Data Mining*, Columbus, OH, May 2010.

[31] Panagiotis Moutafis, Mengjun Leng, and Ioannis A. Kakadiaris, "An overview and empirical comparison of distance metric learning methods," *IEEE Transactions on Cybernetics*, vol. 47, no. 3, pp. 612–625, 2017.

[32] A. V. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," *SIAM Journal on Scientific Computing*, vol. 23, no. 2, pp. 517–541, 2001.

[33] "CVX Research," `http://cvxr.com/cvx/`, Accessed: 2021-5-28.

[34] "CDCS implementation," `https://github.com/oxfordcontrol/CDCS`, Accessed: 2021-5-28.

[35] "SDcut implementation," `https://github.com/chhshen/SDCut`, Accessed: 2021-5-28.

[36] C. Zhu, R. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, Dec. 1997.

[37] "BCR implementation," `https://github.com/shahsohil/biconvex-relaxation`, Accessed: 2021-5-28.

[38] "UCI machine learning repository," `https://archive.ics.uci.edu/ml/datasets.php`, Accessed: 2021-5-28.

[39] "LibSVM Data: Classification (Binary Class)," `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`, Accessed: 2021-5-28.

[40] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall Press, USA, 3rd edition, 2009.

[41] M. Dong, Y. Wang, X. Yang, and J. Xue, "Learning local metrics and influential regions for classification," *IEEE TPAMI*, vol. 42, no. 6, pp. 1522–1529, June 2020.

Checklist

1. For all authors...

(a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] . The paper proposes a new fast algorithm for the precisely defined graph-based classifier learning problem.

(b) Did you describe the limitations of your work? [Yes] . In the conclusion, we discussed the limitation of our work: graph-based binary classification is somewhat narrowly defined, compared to the more general *semi-definite programming* (SDP) problem. However, we conjecture that similar optimization strategies can be customized for other SDP problems, which is left for future work. Moreover, the performance of a graph classifier depends heavily on the construction of a similarity graph, which is outside the scope of this paper.

(c) Did you discuss any potential negative societal impacts of your work? [Yes] . In the conclusion, we discussed potential misuse of graph classifiers that may result in discriminatory classification.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes] Assumptions for the original SDP primal problem (8) are stated in Section 4.1. Assumptions for Lemma 1 are stated in Section 4.3.

(b) Did you include complete proofs of all theoretical results? [Yes] Proof of Lemma 1 is provided in Section 4.3.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code, data, and instructions needed to reproduce the main experimental results are available at the link provided in footnote 4 of Section 6.1.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The convergence thresholds and maximum number of iterations of the core algorithms (if any) in each evaluated method were described in paragraph 2 of Section 6.1.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We reported the average runtime of each problem for each evaluated method in Fig. 4 of Section 6. We reported the type of resources used in paragraph 2 of Section 6.1.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We included the original papers and URL's that produced the code packages in paragraph 1 of Section 6.1. We included the URL's where the datasets are freely available in paragraph 3 of Section 6.1.

(b) Did you mention the license of the assets? [Yes] We included the license of the code packages used in our experiments in paragraph 1 of Section 6.1.

(c) Did you include any new assets either in the supplemental material or as a URL? [No] We included all experimented assets in the main body of our paper.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We described the datasets used in the experiments, which are freely available in the URL's we provided in [38] and [39] of Section 6.1.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]