APPENDIX A
ACCELERATED PROJECTED GRADIENT ALGORITHM

In this section, we provide a detailed explanation on the accelerated projected gradient (APG) method employed in Section 5. The APG method is a first-order method based on the acceleration scheme developed by Nesterov [34]; it has been applied for solving various machine learning algorithms (formulations) [45].

To solve the specific mathematical formulation in Eq. (20), APG maintains a feasible solution sequence $\{Z_i\}$ and a searching point sequence $\{S_i\}$ by recycling the following two steps:

- Update the searching point S_i as below:
$$S_i = (1+\alpha_i)Z_i - \alpha_i Z_{i-1}. \quad (35)$$
- Update the solution point Z_{i+1} as below: (1) compute an auxiliary point $\hat{Z}$ via solving
$$\hat{Z} = \arg\min_{Z\in\mathcal{C}} \frac{\gamma_i}{2}\left\| Z - \left(S_i - \frac{1}{\gamma_i}\nabla f(S_i)\right)\right\|_F^2 + g(Z); \quad (36)$$
(2) set $Z_{i+1} = \hat{Z}$ if $\hat{Z}$ satisfies the inequality
$$f(\hat{Z}) \le f(S_i) + \left\langle \nabla f(S_i), \hat{Z} - S_i\right\rangle + \frac{\gamma_i}{2}\|\hat{Z} - S_i\|_F^2; \quad (37)$$
otherwise set $\gamma_i = 2\times\gamma_i$ and re-compute $\hat{Z}$ via Eq. (36).

The computation in Eq. (36) and Eq. (37) are two main components of the APG method; the former one is commonly referred to as the proximal operator [44], which is involved in each iteration of APG; the latter one (verification of the inequality in terms of γ_i) is referred to as line search (for an appropriate step size estimation).

In the following presentation, we illustrate the computation of the proximal operator, the line search, the main algorithm as well as the convergence analysis. Note that we assume that the smooth convex component $f(\cdot)$ in Eq. (20) has a Lipschitz constant L_f [55]:
$$\|\nabla f(Z) - \nabla f(S)\|_F \le L_f\|Z - S\|_F, \forall S, Z \in \mathcal{C}; \quad (38)$$
the smallest Lipschitz constant $\hat{L}_f$ in Eq. (38), $\hat{L}_f = \min L_f$, is called the best Lipschitz constant for the function $f(\cdot)$.

Proximal Operator Computation Consider the following construction
$$f_L(S, Z) = f(S) + \langle Z - S, \nabla f(S)\rangle + \frac{L}{2}\|Z - S\|_F^2. \quad (39)$$
Clearly $f_L(S, Z)$ is strongly convex with respect to the variable Z. For any $L \ge \hat{L}_f$, it can be verified that the inequality
$$f(Z) \le f_L(S, Z) \quad (40)$$
holds [34]. Denote
$$G_L(S, Z) = f_L(S, Z) + g(Z), \quad (41)$$
where $f_L(S, Z)$ is defined in Eq. (39) and $g(Z)$ is the non-smooth component of the objective function in Eq. (20). Denote by $Z_{L,S}$ the global minimizer to $G_L(S, Z)$ with respect to Z, i.e.,
$$Z_{L,S} = \arg\min_{Z\in\mathcal{M}} G_L(S, Z). \quad (42)$$
It can be verified that $Z_{L,S}$ is unique, as $f_L(Z, S)$ is strongly convex and $g(Z)$ is convex with respect to Z; moreover, $Z_{L,S}$ can be obtained via solving the proximal operator as
$$\min_{Z\in\mathcal{C}} \frac{L}{2}\left\| Z - \left(S - \frac{1}{L}\nabla f(S)\right)\right\|_F^2 + g(Z).$$
Note that the efficient computation of the proximal operator is important for the practical algorithm efficiency, as it is involved in each iteration of the APG method.

Step Size Estimation Given L and S, denote
$$\mathcal{P}_L(S) = L(S - Z_{L,S}). \quad (43)$$
$\mathcal{P}_L(S)$ is called the projected gradient of $f(\cdot)$ at S. By rewriting Eq. (43) as
$$Z_{L,S} = S - \frac{1}{L}\mathcal{P}_L(S), \quad (44)$$
clearly $1/L$ can be seen as the step size associated with the projected gradient $\mathcal{P}_L(S)$.

Denote the objective function in Eq. (20) as
$$F(Z) = f(Z) + g(Z). \quad (45)$$
It follows from Eq. (40) that
$$F(Z_{L,S}) \le G_L(S, Z_{L,S}),\ \forall L \ge \hat{L}_f. \quad (46)$$
If the inequality in Eq. (46) is satisfied, $\mathcal{P}_L(S)$ is called the L-gradient of f at S [33]. In practice we can estimate an appropriate L (hence the appropriate step size $1/L$) by ensuring the inequality in Eq. (46). Theoretically, any step size $1/L$ of the value L larger than the best Lipschitz constant $\hat{L}_f$ guarantees the global convergence in the projected gradient based algorithms [33].

Main Algorithm The pseudo-codes of the APG method are presented in Algorithm 1. The parameters α_i and t_i are auxiliary variables which are constructed to maintain the solution point sequence $\{Z_i\}$ and the searching point sequence $\{S_i\}$; they are important for the convergence analysis of APG. Clearly the solution point Z_i is always feasible in Eq. (20); it converges to the globally optimal solution of Eq. (20). On the other hand, the searching point S_i is not necessarily feasible; S_i can be seen as a forecast of the next feasible solution point Z_{i+1}, as it is a linear combination of two feasible solution points from previous iterations.

Convergence Analysis Using standard techniques in [33], [34], we can show that the APG method in

Algorithm 1 Solving Eq. (20) via the APG Method

1: **Input:** Z_0, $L_0 \in \mathbb{R}$, and max-iter.
2: **Output:** Z.
3: Set $Z_1 = Z_0$, $t_{-1} = 0$, and $t_0 = 1$.
4: **for** $i = 1, 2, \cdots,$ max-iter **do**
5: Compute $\alpha_i = (t_{i-2} - 1)/t_{i-1}$.
6: Compute $S = (1 + \alpha_i)Z_i - \alpha_i Z_{i-1}$.
7: **while** (**true**)
8: Compute $\hat{Z} = Z_{L_i,S}$ via Eq. (42).
9: **if** $F(\hat{Z}) \leq G_{L_i}(S, \hat{Z})$ **then** exit the loop
10: **else** update $L_i = L_i \times 2$.
11: **end-if**
12: **end-while**
13: Update $Z_{i+1} = \hat{Z}$ and $L_{i+1} = L_i$.
14: **if** the stopping criterion is satisfied **then** exit the loop.
15: Update $t_i = \frac{1}{2}(1 + \sqrt{1 + 4t_{i-1}^2})$.
16: **end-for**
17: Set $Z = Z_{i+1}$.

Algorithm 1 globally converges at rate of $\mathcal{O}(1/k^2)$, where k denotes the iteration number. For completeness, we present the detailed convergence analysis of APG as below. Before presenting the detailed convergence analysis, we first establish an important inequality [56], [33], as summarized in the following lemma.

Lemma A.1. *Let L_f be the Lipschitz continuous gradient associated with the function $f(\cdot)$ as defined in Eq. (38). Let $Z_{L,S}$ be the minimizer of $G_L(S, Z)$ as defined in Eq. (42). Then if $L \geq L_f$, the following inequality holds*

$$F(Z) - F(Z_{L,S}) \geq \langle Z - S, \mathcal{P}_L(S)\rangle + \frac{1}{2L}\|\mathcal{P}_L(S)\|_F^2 \quad (47)$$

for any $Z \in \mathcal{C}$.

Proof: It follows from the convexity of $f(\cdot)$ and $g(\cdot)$ that

$$f(Z) \geq f(S) + \langle Z - S, \nabla f(S)\rangle \quad (48)$$
$$g(Z) \geq g(Z_{L,S}) + \langle Z - Z_{L,S}, \partial g(Z_{L,S})\rangle, \quad (49)$$

where $\partial g(Z_{L,S})$ denotes the subgradient [34] of $g(\cdot)$ at $Z_{L,S}$. It is well known that $\hat{Z}$ minimizes $G_L(S, Z)$ (with respect to the variable Z) if and only if $\mathbf{0}$ is a subgradient of $G_L(S, Z)$ at $\hat{Z}$, that is,

$$\mathbf{0} \in L\,(Z_{L,S} - S) + \nabla f(S) + \partial g(Z_{L,S}). \quad (50)$$

From Eqs. (41), (45), (48) and (49), we have

$$\begin{aligned} & F(Z) - G_L(S, Z_{L,S}) \\ = \; & f(Z) + g(Z) - f_L(S, Z_{L,S}) - g(Z_{L,S}) \\ \geq \; & \langle Z - Z_{L,S}, \nabla f(S) + \partial g(Z_{L,S})\rangle - \frac{L}{2}\|S - Z_{L,S}\|_F^2 \\ = \; & -L\,\langle Z - Z_{L,S}, Z_{L,S} - S\rangle - \frac{L}{2}\|S - Z_{L,S}\|_F^2 \\ = \; & \langle Z - S, \mathcal{P}_L(S)\rangle + \frac{1}{2L}\|\mathcal{P}_L(S)\|_F^2, \end{aligned}$$

where the second and third equalities follow from Eqs. (50) and (43), respectively. This completes the proof of this lemma. □

We present the main convergence rate analysis of APG in the following theorem.

Theorem A.1. *Let Z^* be the global minimizer to Eq. (20); let $\hat{L}_f$ be the best Lipschitz continuous gradient defined in Eq.(38). Denote by k the index of the iteration, and by Z_k the solution point at the kth iteration of Algorithm 1. Then we have*

$$F(Z_{k+1}) - F(Z^*) \leq \frac{2\hat{L}}{k^2}\|Z_0 - Z^*\|_F^2,$$

where $\hat{L} = \max\{L_0, 2\hat{L}_f\}$, where L_0 and Z_0 are the initial values of L_k and Z_k in Algorithm 1, respectively.

Proof: Let Z_i be the intermediate solution point at the ith iteration and denote

$$\varepsilon_i = F(Z_i) - F(Z^*).$$

Setting $Z = Z_i$, $S = S_i$, and $L = L_i$ in Eq. (47), we have

$$\epsilon_i - \epsilon_{i+1} \geq \langle Z_i - S_i, \mathcal{P}_{L_i}(S_i)\rangle + \frac{1}{2L_i}\|\mathcal{P}_{L_i}(S_i)\|_F^2, \quad (51)$$

where the left side of the inequality above follows from

$$Z_{i+1} = Z_{L_i,S_i} = \arg\min_{Z \in \mathcal{C}} G_{L_i}(S_i, Z_i).$$

Similarly, setting $Z = Z^*$, $S = S_i$, and $L = L_i$ in Eq. (47), we have

$$-\epsilon_{i+1} \geq \langle Z^* - S_i, \mathcal{P}_{L_i}(S_i)\rangle + \frac{1}{2L_i}\|\mathcal{P}_{L_i}(S_i)\|_F^2. \quad (52)$$

Multiplying Eq. (51) by $t_{i-1} - 1$ and summing it with Eq. (52), we have

$$\begin{aligned} (t_{i-1} - 1)\,\varepsilon_i - t_{i-1}\varepsilon_{i+1} \geq \; & \langle (t_{i-1} - 1)(Z_i - S_i) + Z^* \\ & - S_i, \mathcal{P}_{L_i}(S_i)\rangle + \frac{t_{i-1}}{2L_i}\|\mathcal{P}_{L_i}(S_i)\|_F^2. \end{aligned} \quad (53)$$

Moreover, multiplying Eq. (53) by t_{i-1}, we have

$$\begin{aligned} t_{i-2}^2\varepsilon_i - t_{i-1}^2\varepsilon_{i+1} \geq \; & \frac{1}{2L_i}\|t_{i-1}\mathcal{P}_{L_i}(S_i)\|_F^2 \\ & + \langle t_{i-1}\mathcal{P}_{L_i}(S_i), (t_{i-1} - 1)(Z_i - S_i) + Z^* - S_i\rangle. \end{aligned} \quad (54)$$

where the left side is obtained via the equation

$$t_{i-1}^2 - t_{i-1} = t_{i-2}^2$$

from the line 15 in Algorithm 1. On the other hand, it follows from Eq. (43) we have

$$\mathcal{P}_{L_i}(S_i) = L_i\,(S_i - Z_{L_i,S_i}) = L_i\,(S_i - Z_{i+1})\,. \quad (55)$$

From Eq. (35) and the line 5 in Algorithm 1, we have

$$t_{i-1}S_i = t_{i-1}Z_i + (t_{i-2} - 1)(Z_i - Z_{i-1}). \quad (56)$$

Denote

$$C_{i-2} = t_{i-2}Z_i - (t_{i-2} - 1)Z_{i-1} - Z^*. \quad (57)$$

From Eqs. (55), (56) and (57), we can verify that

$$t_{i-1}\mathcal{P}_{L_i}(S_i)=t_{i-1}L_i(S_i-Z_{i+1})=L_i(C_{i-2}-C_{i-1}). \quad (58)$$

Moreover, we have

$$\begin{aligned}&(t_{i-1}-1)(Z_i-S_i)+Z^*-S_i\\ =\;&(t_{i-1}-1)Z_i+Z^*-t_{i-1}S_i\\ =\;&-t_{i-2}Z_i+(t_{i-2}-1)Z_{i-1}+Z^*=-C_{i-2}.\end{aligned} \quad (59)$$

Substituting Eqs. (58) and (59) into Eq. (54), we obtain

$$\begin{aligned}\|C_{i-1}\|_F^2-\|C_{i-2}\|_F^2 &\le \frac{2}{L_i}\left(t_{i-2}^2\varepsilon_i-t_{i-1}^2\varepsilon_{i+1}\right)\\ &\le \frac{2}{L_{i-1}}t_{i-2}^2\varepsilon_i-\frac{2}{L_i}t_{i-1}^2\varepsilon_{i+1}.\end{aligned} \quad (60)$$

Summing Eq. (60) from $i=1$ to $i=k$, we have

$$\|C_{k-1}\|_F^2-\|C_{-1}\|_F^2 \le \frac{2}{L_0}t_{-1}^2\varepsilon_1-\frac{2}{L_k}t_{k-1}^2\varepsilon_{k+1}.$$

Therefore, we have

$$\begin{aligned}\frac{2}{L_k}t_{k-1}^2\varepsilon_{k+1} &\le \|C_{-1}\|_F^2-\|C_{k-1}\|_F^2+\frac{2}{L_0}t_{-1}^2\varepsilon_1\\ &\le \|C_{-1}\|_F^2+\frac{2}{L_0}t_{-1}^2\varepsilon_1=\|Z_0-Z^*\|^2,\end{aligned} \quad (61)$$

where the equality follows from $t_{-1}=0$ in Algorithm 1. From line 15 in Algorithm 1, we have

$$2t_i=1+\sqrt{1+4t_{i-1}^2}\ge 2t_{i-1}+1. \quad (62)$$

Summing Eq. (62) from $i=1$ to $i=k$, we have

$$t_k\ge\frac{1}{2}(k+1),\quad \forall k. \quad (63)$$

Substituting Eq. (63) into Eq. (61), we complete the proof. □

APPENDIX B
EXAMPLE: MULTI-TASK LEARNING WITH HINGE LOSS FUNCTION

In this section, we present a concrete example to illustrate the BCD method and the APG algorithm for solving rASO in Eq. (8). We employ the hinge loss function in the rASO formulation:

$$\begin{aligned}\min_{\{u_\ell\},M}\quad &\sum_{\ell=1}^m\frac{1}{n_\ell}\sum_{i=1}^{n_\ell}L\left(u_\ell^Tx_i^\ell,y_i^\ell\right)+c\;G_2(U,M)\\ \text{subject to}\quad &\text{tr}(M)=h,\;\mathbf{0}\preceq M\preceq\mathbf{I},\end{aligned} \quad (64)$$

where $G_2(U,M)$ is defined in Eq. (9), and the loss function $L(\cdot)$ is given by

$$L\left(u_\ell^Tx_i^\ell,y_i^\ell\right)=\max\left(1-y_i^\ell\left(u_\ell^Tx_i^\ell+b^\ell\right),0\right),$$

and the parameter c is given by $c=\alpha\eta(1+\eta)$. Note that the optimization problem in Eq. (64) is non-smooth convex due to the non-smooth hinge loss function.

B.1 The BCD Method for Solving Eq. (64)

We employ the BCD algorithm in Section 4 to solve Eq. (64). We focus on discussing its main computational procedures.

Optimization of U Given a fixed M, we can optimize the variable U as

$$\begin{aligned}\min_{\{u_\ell,b_\ell\}}\quad &\sum_{\ell=1}^m\left(\sum_{i=1}^{n_\ell}\xi_{\ell i}+c\;u_\ell^T\left(\eta\mathbf{I}+M\right)^{-1}u_\ell\right)\\ \text{subject to}\quad &\xi_{\ell i}\ge 0,\;\xi_{\ell i}\ge 1-y_i^\ell\left(u_\ell^Tx_i^\ell+b_\ell\right).\end{aligned}$$

Since all pairs of the variables $\{u_\ell,b_\ell\}$ ($\ell\in\mathbb{N}_m$) in the problem above are decoupled, we can optimize each of the pairs by solving a QP problem in the form of

$$\begin{aligned}\min_{u,b}\quad &\sum_{i=1}^n\xi_i+c\;u^T(\eta\mathbf{I}+M)^{-1}u\\ \text{subject to}\quad &\xi_i\ge 0,\xi_i\ge 1-y_i(u^Tx_i+b),i\in\mathbb{N}_n.\end{aligned} \quad (65)$$

The optimal solution to Eq. (65) can be obtained via solving its equivalent primal or dual formulation (in standard SVM form) using existing SVM solvers such as the LIBSVM package [42] as follows. Specifically in Eq. (65), if the involved Hessian matrix $(\eta\mathbf{I}+M)^{-1}$ has a small size such that its SVD can be efficiently computed, the associated optimization problem can be solved via equivalent reformulation into a support vector machine (SVM) as: (1) compute the full SVD $(\eta\mathbf{I}+M)^{-1}=\hat{P}\hat{\Sigma}\hat{P}^T$; (2) set $\hat{u}=\hat{\Sigma}^{\frac{1}{2}}\hat{P}^Tu$ and $\hat{x}_i=\hat{\Sigma}^{-\frac{1}{2}}\hat{P}x_i$; (3) reformulate Eq. (65) into as a standard SVM by variable substitution as

$$\begin{aligned}\min_{\hat{u},b}\quad &\frac{1}{2}\|\hat{u}\|^2+\frac{1}{2c}\sum_{i=1}^n\xi_i\\ \text{subject to}\quad &\xi_i\ge 0,\xi_i\ge 1-y_i(\hat{u}^T\hat{x}_i+b),i\in\mathbb{N}_n.\end{aligned} \quad (66)$$

On the other hand, in Eq. (65), if the involved Hessian matrix has a relatively large size, computing the full SVD on $(\eta\mathbf{I}+M)^{-1}$ may be expensive. For this setting, we convert Eq. (65) into its equivalent dual form, in which the number of optimization variables only depends on the sample size. By augmenting the objective function of Eq. (65) with the constraints, we have the associated Lagrange function as

$$\begin{aligned}L=&\sum_{i=1}^n\xi_i+c\;u^T(\eta\mathbf{I}+M)^{-1}u-\sum_{i=1}^n\alpha_i\xi_i\\ &-\sum_{i=1}^n\beta_i\left(\xi_i-1+y_i(u^Tx_i+b)\right),\end{aligned}$$

where $\alpha_i,\beta_i\ge 0$ denote the dual variables. Taking derivatives with respective to the primal variables

ξ_i, u, b and setting them equal to zero, we have

$$\frac{\partial L}{\partial \xi_i} = 1 - \alpha_i - \beta_i = 0, \quad (67)$$

$$\frac{\partial L}{\partial u} = 2c\ (\eta \mathbf{I} + M)^{-1} u - \sum_{i=1}^{n} \beta_i y_i x_i = 0, \quad (68)$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \beta_i y_i = 0. \quad (69)$$

By substituting Eqs. (67), (68),and (69) into Eq. (65), we have the dual optimization problem as

$$\begin{aligned} \min_{u,b} \quad & \beta^T e - \frac{1}{2} \beta^T \text{ diag}(y) \text{ Ker diag}(y)\ \beta \\ \text{subject to} \quad & 0 \preceq \beta \preceq 1,\ \beta^T y = 0, \end{aligned} \quad (70)$$

where Ker $= \frac{1}{2c} X^T (\eta \mathbf{I} + M) X \in \mathbb{R}^{n \times n}$. Similarly, the optimization problem above is the dual form of a standard SVM with the kernel matrix Ker.

Optimization of M Given a fixed U, we can optimize the variable M as

$$\begin{aligned} \min_{M} \quad & \text{tr}\left(U^T (\eta \mathbf{I} + M)^{-1} U \right) \\ \text{subject to} \quad & \text{tr}(M) = h,\ \mathbf{0} \preceq M \preceq \mathbf{I}. \end{aligned}$$

The optimal M_Z can be obtained via two steps:

- **Step** 1 Compute the SVD of U as $U = P_1 \Sigma P_2^T$, where $P_1 \in \mathbb{R}^{d \times d}$ and $P_2 \in \mathbb{R}^{m \times m}$ are orthogonal, and $\Sigma = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_q, 0, \cdots, 0) \in \mathbb{R}^{d \times m}$ has q non-zero singular values on its main diagonal.
- **Step** 2 Solve the optimization problem as

$$\begin{aligned} \min_{\{\gamma_i\}_{i=1}^q} \quad & \sum_{i=1}^{q} \frac{\sigma_i^2}{\eta + \gamma_i} \\ \text{subject to} \quad & \sum_{i=1}^{q} \gamma_i = h,\ 0 \le \gamma_i \le 1. \end{aligned} \quad (71)$$

and denote its optimal solution by $\{\gamma_i^*\}$.

The optimal M to Eq. (71) is given by $M = P_1 \Lambda^* P_1^T$, where $\Lambda^* = \text{diag}(\lambda_1^*, \cdots, \lambda_q^*, 0, \cdots, 0) \in \mathbb{R}^{d \times m}$.

The Main BCD Algorithm The pseudo-codes of the BCD method for solving Eq. (64) is presented in Algorithm 2. Note the convergence criterion in line 10 can be set as follows. The change of objective values in two successive steps is smaller than a prespecified value (e.g., 10^{-6}).

B.2 The APG Algorithm for Solving Eq. (64)

We employ the APG algorithm in Section 5 to solve Eq. (64). Similarly, we focus on discussing the efficient algorithms for solving the key computational procedures involved in APG.

Algorithm 2 Solve Eq. (64) via the BCD Algorithm

1: **Input:** $\{(x_i^\ell, y_i^\ell)\}$, $i \in \mathbb{N}_{n_\ell}$, $\ell \in \mathbb{N}_m$, $h \in \mathbb{N}$.
2: **Output:** U, V, and M.
3: **Parameter:** α and β.
4: Initialize M subject to the constraints in Eq. (64).
5: **repeat**
6: Update U via Eq. (70).
7: Compute the SVD $U = P_1 \Sigma P_2^T$.
8: Compute $\{\gamma_i^*\}_{i=1}^q$ via Eq. (71).
9: Update M as $M = P_1 \text{diag}(\gamma_1^*, \cdots, \gamma_q^*) P_1^T$.
10: **until** convergence criterion is satisfied.
11: Construct Θ using the top h eigenvectors of M.
12: Construct V as $V = \Theta U$.
13: Return U, V and Θ.

B.2.1 Proximal Operator Computation

To compute the optimal solution Eq. (64), APG iteratively solves the associated proximal operator, i.e., an optimization problem in the general form as

$$\begin{aligned} \min_{U,M} \quad & \|U - \hat{U}\|_F^2 + \|M - \hat{M}\|_F^2 + \hat{\gamma} \sum_{\ell=1}^{m} \sum_{i=1}^{n_\ell} \xi_{\ell i} \\ \text{subject to} \quad & \xi_{\ell i} \ge 0,\ \xi_{\ell i} \ge 1 - y_i^\ell (u_\ell^T x_i^\ell + b_\ell), \\ & \text{tr}(M) = h,\ \mathbf{0} \preceq M \preceq \mathbf{I}, \end{aligned} \quad (72)$$

where $\hat{\gamma} = \frac{2}{\gamma}$. In Eq. (72), the optimization variables U and M are decoupled and the optimal solution can be obtained independently via solving two optimization problems as below.

Computation of U The optimal U to Eq. (72) can be computed via solving

$$\begin{aligned} \min_{U_Z} \quad & \|U - \hat{U}\|_F^2 + \hat{\gamma} \sum_{\ell=1}^{m} \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & \xi_{\ell i} \ge 0,\ \xi_{\ell i} \ge 1 - y_i^\ell (u_\ell^T x_i^\ell + b_\ell). \end{aligned} \quad (73)$$

Let $U = [u_1 \cdots u_m]$ and $\hat{U} = [\hat{u}_1 \cdots \hat{u}_m]$. Each of the vector u_ℓ can be obtained by solving a QP problem as

$$\begin{aligned} \min_{u_\ell} \quad & \|u_\ell - \hat{u}_\ell\|_F^2 + \hat{\gamma} \sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & \xi_{\ell i} \ge 0,\ \xi_{\ell i} \ge 1 - y_i^\ell (u_\ell^T x_i^\ell + b_\ell). \end{aligned} \quad (74)$$

The QP problem in Eq. (74) is similar to the standard SVM formulation, while $\|u_\ell - \hat{u}_\ell\|_F^2$ leads to a linear term $2\hat{u}_\ell^T u_\ell$ in its primal and dual formulations; hence this QP problem could not be solved directly using the existing SVM solver. We can solve this QP problem using the general optimization solvers such as MOSEK[3], the sequential minimal optimization (SMO) algorithm [57], or the standard projected conjugate gradient (PCG) chunking algorithm. Note that both SMO and PCG avoid the large matrix computation; SMO scales between linearly and quadratically with

3. http://www.mosek.com/

the training size while PCG scales between linearly and cubically with the training size [57].

Computation of M The optimal M to Eq. (72) can be computed via solving

$$\begin{array}{ll}\min\limits_{M_Z} & \|M-\hat{M}\|_F^2 \\ \text{subject to} & \text{tr}(M)=h,\ \mathbf{0}\preceq M\preceq \mathbf{I}. \end{array} \quad (75)$$

The optimal M can be obtained via two steps:

- **Step** 1 Compute the eigendecomposition of the symmetric $\hat{M}$ as $\hat{M}=P\hat{\Sigma}P^T$, where $P\in\mathbb{R}^{m\times m}$ is orthogonal, $\hat{\Sigma}=\text{diag}(\sigma_1,\cdots,\sigma_m)\in\mathbb{R}^{m\times m}$ is diagonal with the eigenvalues on its main diagonal.
- **Step** 2 Solve the optimization problem

$$\begin{array}{ll}\min\limits_{\{\sigma_i\}} & \sum\limits_{i=1}^{m}(\sigma_i-\hat{\sigma}_i)^2 \\ \text{subject to} & \sum\limits_{i=1}^{m}\sigma_i=h,\ 0\le\sigma_i\le 1. \end{array}$$

and denote its optimal solution by $\{\sigma_i^*\}$.

The optimal M to Eq. (75) is given by $M=P\Sigma^*P^T$, where $\Sigma^*=\text{diag}(\sigma_1^*,\cdots,\sigma_m^*)\in\mathbb{R}^{m\times m}$.

B.2.2 The Main APG Algorithm

The pseudo-codes of the APG algorithm for solving Eq. (64) are similar to Algorithm 1. Specifically to solve Eq. (64), APG solves Eq. (72) (the proximal operator) in each of its iteration, where $\hat{\gamma}$ is set as $2/L_i$ and the value of L_i is determined via the line search scheme in Eq. (37).

APPENDIX C
SOLVING EQ. (14) IN LINEAR TIME

We present a parametric approach to solve the optimization problem in Eq. (14) in linear time. This approach is previously proposed in [43] to solve a bounded quadratic optimization problem with multiple variables and one linear constraint.

C.1 A Parametric Formulation

We consider a parametric variant of Eq. (14) as

$$\begin{array}{ll}\min\limits_{\{\gamma_i\}_{i=1}^q} & \sum\limits_{i=1}^{q}\left(\dfrac{\sigma_i^2}{\eta+\gamma_i}+t\gamma_i\right) \\ \text{subject to} & 0\le\gamma_i\le 1,\ i\in\mathbb{N}_q, \end{array} \quad (76)$$

where t is an arbitrary parameter. Obviously for a given t, the optimization of $\{\gamma_i\}_{i=1}^q$ in Eq. (76) is decoupled; each optimal γ_i can be obtained via solving

$$\begin{array}{ll}\min\limits_{\gamma_i} & \dfrac{\sigma_i^2}{\eta+\gamma_i}+t\gamma_i \\ \text{subject to} & 0\le\gamma_i\le 1. \end{array} \quad (77)$$

The optimization problem above is convex and its objective function is piecewise differentiable. For clear presentation, we denote the optimal solution to Eq. (77) by $\gamma_i(t)$. It can be verified that

$$\gamma_i(t)=\begin{cases}0 & \frac{\sigma_i}{\sqrt{t}}-\eta\le 0\\ \frac{\sigma_i}{\sqrt{t}}-\eta & 0<\frac{\sigma_i}{\sqrt{t}}-\eta<1\\ 1 & 1\le\frac{\sigma_i}{\sqrt{t}}-\eta\end{cases}. \quad (78)$$

C.2 Computing an Optimal Solution with the Appropriate Interval

Define an auxiliary function $\Gamma(t)$ as

$$\Gamma(t)=\sum_{i=1}^{q}\gamma_i(t). \quad (79)$$

Using the techniques from [43], we can verify several important properties of $\Gamma(t)$, as summarized in the following theorem (proof omitted).

Theorem C.1. *Let $\Gamma(t)$ be defined in Eq. (79). Then it satisfies the following properties.*

(1) *$\Gamma(t)$ is monotone not increasing in terms of t.*
(2) *If $\Gamma(t)=h$, then $\{\gamma_i(t)\}_{i=1}^q$ is optimal to Eq. (14).*
(3) *If $\Gamma(t)\neq h$ for all $t\in\mathbb{R}$, no feasible solution exists in Eq. (14).*

Definite the critical parameter pairs t_i^l and t_i^u [43] by

$$t_i^l=\frac{\sigma_i^2}{(\eta+1)^2},\quad t_i^u=\frac{\sigma_i^2}{\eta^2}, \quad (80)$$

and denote all distinct critical parameters by $t_1,t_2,\cdots,t_r$, where $t_1<t_2<\cdots<t_r$. It follows from Eq. (78) that within any interval (t_i,t_{i+1}), $\gamma_i(t)$ has a unique expression. Moreover, from Theorem C.1, we can verify that if $\Gamma(t_1)>h$ and $\Gamma(t_r)<h$, there exists an index j such that $\Gamma(t_j)\ge h$ and $\Gamma(t_{j+1})\le h$. Note that the interval (t_j,t_{j+1}) is referred to as the appropriate interval for Eq. (14).

Given the appropriate interval, the optimal solution to Eq. (14) can be constructed as follows. Define three index sets: $I_l=\{i\,|\,t_i^l\le t_j\}$, $I_u=\{i\,|\,t_{j+1}\le t_i^u\}$, and $I_m=\{i\,|\,t_j<t_i^u,t_i^l<t_{j+1}\}$. For any arbitrary $t\in[t_j,t_{j+1}]$, the optimal $\gamma_i(t)$ to Eq. (14) can be expressed as

$$\gamma_i(t)=\begin{cases}0 & i\in I_l\\ \frac{\sigma_i}{\sqrt{t}}-\eta & i\in I_m\\ 1 & i\in I_u\end{cases}. \quad (81)$$

Denote $\Gamma(t_{\text{opt}})=h$. Therefore t_{opt} can be obtained via solving

$$\begin{aligned}\Gamma(t_{\text{opt}}) &= \sum_{i=1}^{q}\gamma_i(t_{\text{opt}})\\ &= \sum_{i\in I_l}0+\sum_{i\in I_m}\left(\frac{\sigma_i}{\sqrt{t}}-\eta\right)+\sum_{i\in I_u}1=h.\end{aligned} \quad (82)$$

By substituting t_{opt} into Eq. (81), we can get the optimal solution to Eq. (14).

Algorithm 3 The Linear Time Search Algorithm

1: **Input:** $\{t_1, \cdots, t_r\}$, $\big\{(t_1^l, t_1^u), \cdots, (t_q^l, t_q^u)\big\}$, and h.
2: **Output:** t_{opt}, $t_{\min}$, and $t_{\max}$.
3: **if** $\Gamma(t_1) = h$ or $\Gamma(t_r) = h$ **then**
4: set $t_{\text{opt}} = t_1$ or $t_{\text{opt}} = t_r$; exit the Algorithm.
5: **end if**
6: **if** $\Gamma(t_1) < h$ or $\Gamma(t_r) > h$ **then**
7: no feasible solution exists; exit the Algorithm.
8: **end if**
9: Set $t_{\min} = t_1$, $t_{\max} = t_r$, and $I = \{1, \cdots, q\}$.
10: **repeat**
11: $t^l = \text{median}\big(\{t_i^l \mid i \in I\}\big)$.
12: $t^u = \text{median}\big(\{t_i^u \mid i \in I; t_i^l \geq t^l\}\big)$.
13: **for** $t = t^l, t^u$ **do**
14: **if** $\Gamma(t) = h$ **then**
15: $t_{\text{opt}} = t$; exit the Algorithm.
16: **end if**
17: **if** $z(t) > h$ **then**
18: $t_{\min} = \max(t_{\min}, t)$.
19: **end if**
20: **if** $z(t) < h$ **then**
21: $t_{\min} = \min(t_{\max}, t)$.
22: **end if**
23: **end for**
24: **for** $i \in I$ **do**
25: **if** $t_i^l \leq t_{\min}$ **then**
26: $I = I \setminus \{i\}$; $\gamma_i = 0$.
27: **end if**
28: **if** $t_{\max} \leq t_i^u$ **then**
29: $I = I \setminus \{i\}$; $\gamma_i = 1$.
30: **end if**
31: **if** $t_i^u \leq t_{\min} \leq t_{\max} \leq t_i^l$ **then**
32: $I = I \setminus \{i\}$; $\gamma_i = \sigma_i / t - \eta$.
33: **end if**
34: **end for**
35: **until** I is an empty set.

C.3 Search for the Appropriate j

Computation of the optimal solution to Eq. (14) boils down to the efficient search of the appropriate interval (t_j, t_{j+1}) from the distinct critical parameters. We employ an efficient algorithm to search the appropriate index j. This algorithm is originally proposed in [43] with linear time complexity.

We present the main idea of the efficient search algorithm below for completeness. Recall that the distinct critical parameters satisfy $t_1 < t_2 < \cdots < t_r$ are . Assume that $\Gamma(t_1) > h$ and $\Gamma(t_r) < h$. From Theorem C.1, we have $t_{\text{opt}} \in [t_1, t_r]$. The general scheme of the linear time search algorithm is as follows. Starting by setting $t_{\min} = t_1$ and $t_{\max} = t_r$, we iteratively update $t_{\min}$ and $t_{\max}$ and meanwhile ensure

$$t_{\text{opt}} \in [t_{\min}, t_{\max}]. \tag{83}$$

For a specific t, if $\Gamma(t) > h$ we set $t_{\min} = \max(t_{\min}, t)$; if $\Gamma(t) < h$, we set $t_{\min} = \min(t_{\max}, t)$; if $\Gamma(t) = h$, we set $t_{\text{opt}} = t$ and exit the algorithm. The pseudocodes of the algorithm is presented in Algorithm 3. Note that in Algorithm 3, the codes from line 24 to line 34 ensures the size of the set I can be reduced by at least $\frac{1}{4}\lceil I \rceil$ elements. It can be verified that the complexity of Algorithm 3 is $\mathcal{O}(q)$, where q denotes the size of the search sequence; detailed analysis can be found in [43]. If Algorithm 3 stops at line 7, no feasible solution exists in Eq. (14); if Algorithm 3 stops at either line 4 or line 15, the optimal solution to Eq. (14) can be obtained by substituting t_{opt} into Eq. (81); otherwise we have $t_j = t_{\min}$ and $t_{j+1} = t_{\max}$; the optimal solution to Eq. (14) can be obtained from Eqs. (81) and (82). Note that before applying the linear time search algorithm, we pre-compute all the critical parameters defined in Eq. (80).