Learning graph structure for multi-label image classification via clique generation

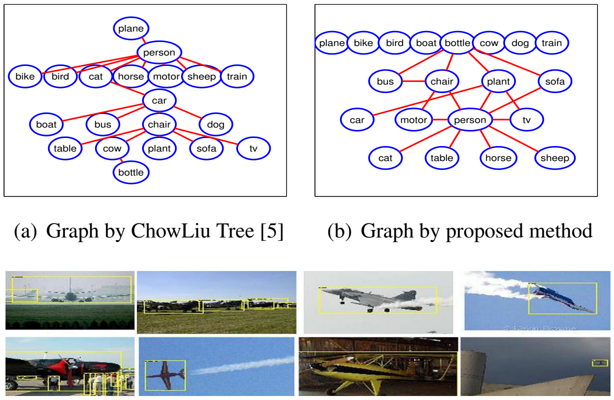
Mingkui Tan[1], Qinfeng Shi[1]*, Anton van den Hengel[1,3]*, Chunhua Shen[1,3]*, Junbin Gao[2],
Fuyuan Hu[1], Zhen Zhang[1]

[1] University of Adelaide [2] Charles Sturt University [3] Australian Centre for Robotic Vision

e-mail: {mingkui.tan,javen.shi}@adelaide.edu.au

Abstract

Exploiting label dependency for multi-label image classification can significantly improve classification performance. Probabilistic Graphical Models are one of the primary methods for representing such dependencies. The structure of graphical models, however, is either determined heuristically or learned from very limited information. Moreover, neither of these approaches scales well to large or complex graphs. We propose a principled way to learn the structure of a graphical model by considering input features and labels, together with loss functions. We formulate this problem into a max-margin framework initially, and then transform it into a convex programming problem. Finally, we propose a highly scalable procedure that activates a set of cliques iteratively. Our approach exhibits both strong theoretical properties and a significant performance improvement over state-of-the-art methods on both synthetic and real-world data sets.



(a) Graph by ChowLiu Tree [5]

(b) Graph by proposed method

(c) Sample images of "plane" in PASCAL2007 database

Figure 1. Comparison of graphs learned from PASCAL2007. Unlike the graph in Figure (1(a)), the graph learned by proposed method has identified categories not connected with other nodes, e.g., "plane". In fact, from Figure (1(c)), the presence of a "plane" in an image is often independent of other objects (except "sky", which is not a labeled class).

1. Introduction

Multi-label image classification is the problem of predicting a binary label vector, each element of which indicates the presence or absence of a certain object category in an image [1, 3]. Exploiting label dependency can significantly boost classification performance. For example, if an instance of the *ship* category is present in an image, it is very likely that the *water* category is also present. To capture label dependencies, it is common practice to use Probabilistic Graphical Models (PGMs) [17], a standard workhorse for modelling dependencies among random variables.

The dependencies in PGMs are often encoded in a graph structure. However, how to correctly and efficiently estimate such a graph is still a challenging problem. People often use heuristics, such as manually specified graph structures based on domain knowledge, or simple rules like the ChowLiu Tree [5, 2] which uses mutual information between labels but ignores features or visual contents of images completely.

We propose to learn the graph structure and parameters by considering both features and labels. In this way, the learned graph structure and parameters will fit the data better. An important concern for graph structure learning is that, while it is essential to find those relevant dependencies, it is also important to identify the *independent labels* and *irrelevant* dependencies. In Figure 1, we show the graph constructed using ChowLiu Trees [5] (Figure 1(a)) and that learned by our method (Figure 1(b)). Their main difference is that, unlike the ChowLiu Tree graph wherein all the labels are connected, some labels in our graph are isolated which makes learning and inference faster and simpler.

*This work was in part funded by the Data to Decisions Cooperative Research Centre, Australia, ARC Grant DP140102270, National Natural Science Foundation Key Project #61231016, and National Natural Science Foundation of China Grant #61472267.

2. Related Work

For multi-label learning, one may train a binary classifier for each label independently, which is known as the binary relevance (BR) method. This approach is used even in cases where labels are, in fact, dependent [1, 6, 39]. To boost performance, many methods have been proposed to exploit the label dependency, including problem transformation (PT) methods [37, 12, 38], output coding methods [16, 32, 44, 43], PGM-based methods [13, 11, 15, 31, 22, 28], amongst others.

PT methods manually construct auxiliary labels, or data, to capture the label dependency, so that existing classification methods can be applied, such as the label powerset method [37], ranking methods [12], and ensemble methods [38]. However, their ability to identify dependencies is limited when the number of labels becomes large.

Some output coding methods attempt to select codes which preserve, or even exploit, dependencies between labels [16, 32, 26, 32, 27]. For a multi-label problem with many labels, the label vectors are often highly sparse. By exploiting this, Hsu *et al.* [16] propose to compress the labels using a random matrix. After learning regressors on compressed labels, the sparse labels are recovered by compressive sensing techniques. However, this method may not work well for non-sparse labels. In [32], a principal label space transformation (PLST) method is proposed to capture label correlations by projecting them onto a lower dimensional space via PCA. This coding scheme does not consider the features, and thus may not be discriminative. To address this, a maximum margin output coding (MMOC) is proposed in [44], which considers both labels and features.

For PGM-based methods, a ChowLiu Tree is widely used to construct a tree structure based on the mutual information of labels [2]. A maximum spanning tree is constructed using label co-occurrence [22]. In [43], a directed acyclic graph is adopted to capture the joint probability of labels. A chain structure is used in [7]. A random graph ensemble method is studied in [31], which constructs a graph from several random graphs. However, these graph constructions are often separated from the parameter learning process. Learning the graph structure along with the parameters has been considered in [28, 8], but these methods are not scalable. Wu *et al.* exploit a tree structure to capture the label relations [39], but this method is very time consuming. More recently, a new graph sparsity model is proposed in [4], but it is not really a graphical model, and can not capture general label dependencies.

A closely related method is [28], which learns graphs in the fashion of Conditional Random Fields (CRFs) [21] (or log-linear models) using a group ℓ_1-regularisation. However, this method is not scalable and does not allow the flexibility of choosing task-related loss functions.

3. Multi-label and Structured Learning

Throughout the paper we denote the transpose of a vector/matrix by the superscript $^\top$ and the zero vector as $\mathbf{0}$. In addition, $\|\mathbf{v}\|_p$ denotes the ℓ_p-norm of a vector $\mathbf{v}$. For a function $f(\mathbf{x})$, the gradient and subgradient of $f(\mathbf{x})$ at $\mathbf{x}$ are denoted by $\nabla f(\mathbf{x})$ and $\partial f(\mathbf{x})$, respectively.

Given a labeled data set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathcal{X}$ and $\mathbf{y}_i \in \mathcal{Y} = \{0, 1\}^K$, multi-label learning seeks to learn a hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}$ that maps an input $\mathbf{x} \in \mathcal{X}$ to outputs $\mathbf{y} \in \mathcal{Y}$. One approach to this problem is by solving the following *decoding* problem:

$$h(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{y}, \mathbf{x}, \mathbf{w}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} \mathbf{w}^\top \mathbf{\Psi}(\mathbf{y}, \mathbf{x}), \quad (1)$$

where $\mathbf{\Psi} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^m$ is a feature mapping that combines the inputs and outputs, and $\mathbf{w}$ denotes the model parameters to be determined.

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{C}\}$ denote a graph, where $\mathcal{V} = \{1, ..., p\}$ denotes the set of all nodes (*i.e.* labels), and $\mathcal{C} \subset \mathcal{V} \times \mathcal{V}$ denotes a set of cliques (fully connected subgraphs) of order higher than 1, i.e., $\mathcal{C} \bigcap \mathcal{V} = \emptyset$. If we restrict each clique to contain only two nodes, then $\mathcal{C}$ recovers the conventional edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. Note that using $\mathcal{C}$ (instead of $\mathcal{E}$) allows our approach to handle higher order potential functions in undirected or factor PGMs.

Based on graph $\mathcal{G}$, as commonly used $F(\mathbf{y}, \mathbf{x}, \mathbf{w})$ is decomposed by potential functions:

$$F(\mathbf{y}, \mathbf{x}, \mathbf{w}) = \sum_{k\in\mathcal{V}} \mathbf{u}_k^\top \mathbf{\Psi}(y^k, \mathbf{x}) + \sum_{c\in\mathcal{C}} \mathbf{v}_c^\top \mathbf{\Psi}_c(\mathbf{y}^c, \mathbf{x}), \quad (2)$$

where $\mathbf{w}$ concatenates all $\mathbf{u}_k$'s and $\mathbf{v}_c$'s, and the global feature $\mathbf{\Psi}(\mathbf{y}, \mathbf{x})$ concatenates all node features $\mathbf{\Psi}(y^k, \mathbf{x})$ and clique features $\mathbf{\Psi}_c(\mathbf{y}^c, \mathbf{x})$. Once a graph is constructed, the model parameters can be learned by Conditional Random Fields (CRFs) [21] or Structured Support Vector Machines (SSVMs) [36]. In SSVMs, to measure the distance between the prediction $h(\mathbf{x}_i)$ and true outputs $\mathbf{y}_i$, a loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ is used, for example, the Hamming loss: $\triangle(\mathbf{y}, \mathbf{y}_i) = \sum_{k=1}^K 1[y^k \neq y_i^k]$ [36].

4. Learning with Unknown Graph

In our setting, the graph $\mathcal{G}$ is unknown, and we propose to learn it and model parameters jointly from the data. Learning $\mathcal{G}$ can be seen as selecting relevant cliques from all possible cliques, which underpins our representation.

4.1. Representation

Though there might be many potential cliques in $\mathcal{C}$, only a few of them are relevant to the output in the sense that, a label (e.g. an object) is often related to only a small number of other objects. If all labels are exclusive to others, the problem is reduced to a multi-class problem.

To find the relevant cliques, we introduce a binary vector $\boldsymbol{\eta} \in \{0,1\}^q$, where $q = |\mathcal{C}|$, and define

$$\boldsymbol{\Psi}_c(\mathbf{y}^c, \mathbf{x}; \eta_c) = \eta_c \boldsymbol{\Psi}_c(\mathbf{y}^c, \mathbf{x}), \forall c \in \mathcal{C}.$$

Here, a clique c is chosen if $\eta_c = 1$; otherwise it will be dropped. For convenience, we define a new feature mapping associated with $\boldsymbol{\eta}$ as $\boldsymbol{\Psi}_{\mathcal{C}}(\mathbf{y}, \mathbf{x}; \boldsymbol{\eta}) = [\eta_c \boldsymbol{\Psi}_c(\mathbf{y}^c, \mathbf{x})]_{c \in \mathcal{C}}$, and the feature mapping difference as $\Phi^{\mathcal{C}}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i; \boldsymbol{\eta}) = \boldsymbol{\Psi}_{\mathcal{C}}(\mathbf{y}_i, \mathbf{x}_i; \boldsymbol{\eta}) - \boldsymbol{\Psi}_{\mathcal{C}}(\mathbf{y}, \mathbf{x}_i; \boldsymbol{\eta})$. Since we keep all the nodes, we define $\boldsymbol{\Psi}_{\mathcal{V}}(\mathbf{y}, \mathbf{x}) = [\boldsymbol{\Psi}(y^k, \mathbf{x})]_{k \in \mathcal{V}}$, and $\Phi^{\mathcal{V}}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i) = \boldsymbol{\Psi}_{\mathcal{V}}(\mathbf{y}_i, \mathbf{x}_i) - \boldsymbol{\Psi}_{\mathcal{V}}(\mathbf{y}, \mathbf{x}_i)$.

We prefer to select the least number of cliques that would fit the data well, thus we introduce an ℓ_0-norm constraint $||\boldsymbol{\eta}||_0 \leq r$, where r denotes the least number of cliques we intend to select. Here, r may represent our basic prior knowledge about the graph. For example, setting $r = 1$ means that there might be one relevant clique. Nevertheless, as will be shown, our later proposed optimization scheme guarantees to find more cliques (if relevant) even when we set $r = 1$. But if the number of relevant cliques is very large, a larger r may dramatically improve computational efficiency.

Let $\Lambda = \{\boldsymbol{\eta} | \boldsymbol{\eta} \in \{0,1\}^q, ||\boldsymbol{\eta}||_0 \leq r\}$ be the domain of $\boldsymbol{\eta}$, $\mathbf{u}$ be the model parameters w.r.t. the node cliques $\mathcal{V}$, and $\mathbf{v}$ be the model parameters w.r.t. $\mathcal{C}$. Based on the SSVM framework [36], we propose to learn a sparse graph (together with the model parameter $\mathbf{w}$) by solving the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\eta} \in \Lambda} \min_{\mathbf{w} = [\mathbf{u}; \mathbf{v}]} \quad & \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i, \qquad (3) \\ \text{s.t.} \quad & \mathbf{u}^\top \Phi^{\mathcal{V}}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i) + \mathbf{v}^\top \Phi^{\mathcal{C}}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i; \boldsymbol{\eta}) \geq \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i, \\ & \xi_i \geq 0 \ \forall i, \forall \mathbf{y} \in \mathcal{Y} \backslash \mathbf{y}_i, \end{aligned}$$

where $\mathbf{u} = [\mathbf{u}_k]_{k \in \mathcal{V}}$, $\mathbf{v} = [\mathbf{v}_c]_{c \in \mathcal{C}}$ and $\mathbf{w} = [\mathbf{u}; \mathbf{v}]$. This problem is non-convex. However, it is reduced to a standard SSVM problem of $l = n|\mathcal{Y}|$ potential constraints by fixing $\boldsymbol{\eta}$ [36, 20]. Introducing non-negative dual variables $\boldsymbol{\alpha} \in \mathbb{R}^l$, the Lagrange dual of the SSVM problem with fixed $\boldsymbol{\eta}$ can be written as

$$\max_{\boldsymbol{\alpha} \in \mathcal{A}} \quad -\frac{1}{2\lambda}||\mathbf{u}(\boldsymbol{\alpha})||^2 - \frac{1}{2\lambda}||\mathbf{v}(\boldsymbol{\alpha}, \boldsymbol{\eta})||^2 + \mathbf{b}^\top \boldsymbol{\alpha}, \qquad (4)$$

where $\mathcal{A} := \{\boldsymbol{\alpha} \in \mathbb{R}^l | \boldsymbol{\alpha} \geq 0, \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \leq \frac{1}{n}\}$ is the domain of $\boldsymbol{\alpha}$, $\mathbf{b} = [\Delta(\mathbf{y}, \mathbf{y}_i)]_{i \in [n], \mathbf{y} \neq \mathbf{y}_i}$, $\mathbf{u}(\boldsymbol{\alpha}) := \sum_{i=1}^{n} \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \, \Phi^{\mathcal{V}}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i)$, and $\mathbf{v}(\boldsymbol{\alpha}, \boldsymbol{\eta}) := \sum_{i=1}^{n} \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \, \Phi^{\mathcal{C}}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i; \boldsymbol{\eta})$.

Let $g(\boldsymbol{\alpha}, \boldsymbol{\eta}) := -\frac{1}{2\lambda}||\mathbf{u}(\boldsymbol{\alpha})||^2 - \frac{1}{2\lambda}||\mathbf{v}(\boldsymbol{\alpha}, \boldsymbol{\eta})||^2 + \mathbf{b}^\top \boldsymbol{\alpha}$. We then transform problem (3) into a **minimax** problem:

$$\min_{\boldsymbol{\eta} \in \Lambda} \max_{\boldsymbol{\alpha} \in \Omega} \quad g(\boldsymbol{\alpha}, \boldsymbol{\eta}), \qquad (5)$$

This problem is still an integer programming problem. According to the minimax inequality [30], we have

$$\min_{\boldsymbol{\eta} \in \Lambda} \max_{\boldsymbol{\alpha} \in \mathcal{A}} \quad g(\boldsymbol{\alpha}, \boldsymbol{\eta}) \geq \max_{\boldsymbol{\alpha} \in \mathcal{A}} \min_{\boldsymbol{\eta} \in \Lambda} \quad g(\boldsymbol{\alpha}, \boldsymbol{\eta}).$$

Therefore, the latter problem is a **convex** lower bound to (5). In fact, by introducing an additional variable $\theta \in \mathbb{R}$, we can further transform it into a standard convex programming problem:

$$\begin{aligned} \max_{\theta \in \mathbb{R}, \boldsymbol{\alpha} \in \mathcal{A}} \quad & \theta, \qquad (6) \\ \text{s.t.} \quad & \theta \leq g(\boldsymbol{\alpha}, \boldsymbol{\eta}), \quad \forall \boldsymbol{\eta} \in \Lambda. \end{aligned}$$

Therefore, problem (6) is a convex relaxation to (5) as well as the original problem (3). Recall that each feasible $\boldsymbol{\eta} \in \Lambda$ corresponds to a quadratic constraint. Then, problem (6) has exponentially many constraints as there are $\binom{q}{r}$ elements in Λ, making it hard to address directly.

4.2. Clique Generation

Though there are exponentially many constraints in (6), most of them should be inactive at the optimum solution, due to the assumption that the graph $\mathcal{G}$ is sparse. Motivated by this, and the cutting plane approach, we propose Algorithm 1 to address this problem.

Algorithm 1 Clique Generation.

```
1: Initialize $\Lambda_0 = \emptyset$ and $\mathcal{C}_t = \emptyset$. For $i = 1, ..., n$: Compute $\mathbf{y}_i^* = \arg\max_{\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y}_i)$; Let $\mathcal{W}_i^0 = \{\mathbf{y}_i^*\}$; Set $\alpha^0_{i\mathbf{y}^*} = K$, where $K$ is the number of labels. Set $t = 1$.
2: Find $\widehat{\boldsymbol{\eta}}$ by solving (7). Let $\Lambda_t = \Lambda_{t-1} \bigcup \{\widehat{\boldsymbol{\eta}}\}$ and $\Gamma_t = \{c\}_{\widehat{\eta_c} = 1}$.
3: Solve subproblem (8) to update $\boldsymbol{\alpha}^t$ and $\mathcal{W}_i^t$.
4: Let $t = t + 1$ and repeat from Step 2 until convergence.
```

Algorithm 1 iteratively activates a number of cliques (also known as the most-violated constraint) until some conditions are achieved. Specifically, at the tth iteration, we find a most-violated constraint (which corresponds to a particular choice of $\boldsymbol{\eta}$) by solving

$$\widehat{\boldsymbol{\eta}} = \arg\min_{\boldsymbol{\eta}} g(\boldsymbol{\alpha}^{t-1}, \boldsymbol{\eta}), \qquad (7)$$

where $\boldsymbol{\alpha}^{t-1}$ is assumed known. At the initialization stage, e.g. $t = 1$, we need the knowledge of $\boldsymbol{\alpha}^0$ and $\mathcal{W}_i^0$ to address $\widehat{\boldsymbol{\eta}} = \arg\min_{\boldsymbol{\eta}} g(\boldsymbol{\alpha}^{t-1}, \boldsymbol{\eta})$. Since we have not learnt any parameters yet (namely $\mathbf{w} = \mathbf{0}$), the active label $\mathbf{y}_i^*$ can be easily obtained by solving $\mathbf{y}_i^* = \arg\max_{\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y}_i)$. By initializing $\mathcal{W}_i^0 = \{\mathbf{y}_i^*\}$, we can just initialize $\alpha^0_{i\mathbf{y}^*} = K$, where K denote the number of labels.

In general, $\boldsymbol{\alpha}^{t-1}$ can be very high-dimensional (namely, $n|\mathcal{Y}|$), but it is always sparse. This property is quite useful for solving (7). In fact, as problem (7) is only related to nonzero $\alpha_{i\mathbf{y}}$'s, we define a **Working Set** of $\alpha_{i\mathbf{y}}$ as:

$$\mathcal{W}_i = \{\mathbf{y}^* | \alpha_{i\mathbf{y}} > 0, \mathbf{y}^* \in \mathcal{Y}\}.$$

Once $\widehat{\boldsymbol{\eta}}$ is determined, we add into an active constraint set Λ_t ($\Lambda_0 = \emptyset$), and subsequently solve a subproblem w.r.t. constraints defined in Λ_t:

$$\begin{aligned} \max_{\theta \in \mathbb{R}, \boldsymbol{\alpha} \in \mathcal{A}} \quad & \theta, \\ \text{s.t.} \quad & \theta \leq g(\boldsymbol{\alpha}, \boldsymbol{\eta}), \ \ \forall \boldsymbol{\eta} \in \Lambda_t. \end{aligned} \tag{8}$$

Since there are only $|\Lambda_t|$ constraints, the complexity of problem (8) is considerably reduced. After solving this problem, we obtain $\boldsymbol{\alpha}^t$, and update $\mathcal{W}_i^t$ accordingly.

Theorem 1. *[35, 33] Assume that both problems (7) and (8) can be addressed per iteration, then CGM stops after a finite number of iterations with a global solution of (6).*

In the subsequent sections we discuss the optimization of (7), and of (8).

4.3. Finding the Most Violated Constraint

To identify the most-violated constraint, we need to solve problem $\widehat{\boldsymbol{\eta}} = \arg\min_{\boldsymbol{\eta}} g(\boldsymbol{\alpha}^{t-1}, \boldsymbol{\eta})$. It is easy to verify that this problem is equivalent to $\widehat{\boldsymbol{\eta}} = \arg\max_{\boldsymbol{\eta}} ||\mathbf{v}(\boldsymbol{\alpha}, \boldsymbol{\eta})||_2^2$. Note that $\eta_c \in \{0, 1\}$, and $\mathbf{v}(\boldsymbol{\alpha}, \boldsymbol{\eta})$ can be written as $\mathbf{v}(\boldsymbol{\alpha}, \boldsymbol{\eta}) = [\eta_c \mathbf{v}_c(\boldsymbol{\alpha})]_{c \in \mathcal{C}}$, where

$$\mathbf{v}_c(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \sum_{\mathbf{y} \in \mathcal{W}_i} \alpha_{i\mathbf{y}} \ \Phi^{\mathcal{C}}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i; \boldsymbol{\eta}).$$

Then, problem (7) can be addressed by solving the following problem:

$$\max_{\boldsymbol{\eta}} \sum_{c \in \mathcal{C}} \eta_c ||\mathbf{v}_c(\boldsymbol{\alpha})||_2^2, \ \ \text{s.t.} \ \ \sum_{c \in \mathcal{C}} \eta_c \leq r, \eta_c \in \{0, 1\}. \tag{9}$$

This problem is an integer programming problem, but its optimal solution can be easily obtained. Let $s_c = ||\mathbf{v}_c(\boldsymbol{\alpha})||_2^2$ be the score of a clique c. We can first find r cliques with the largest score (e.g. s_c), and then set the corresponding η_j to 1 and the rest to 0. We can easily verify that such a $\boldsymbol{\eta}$ is an optimal solution of (9).

In practice, we do not have to maintain $\widehat{\boldsymbol{\eta}}$ explicitly. In fact, since the nonzero entries in $\widehat{\boldsymbol{\eta}}$ essentially index r new cliques, we only need to record these cliques into a set Γ_t. For convenience, we record all the selected cliques of t iterations into a set $\mathcal{C}_t$, which is updated by $\mathcal{C}_t = \mathcal{C}_{t-1} \cup \Gamma_t$.

Lastly, since each $\widehat{\boldsymbol{\eta}}$ activates at most r active cliques, hereafter we call Algorithm 1 the Clique Generating Machine (CGM).

4.4. Optimization on (8)

Solving problem (8) w.r.t. $\boldsymbol{\alpha}$ directly can be very expensive when the number of instances (namely n) is large. In the following, we propose to solve it through a **Stochastic Proximal Dual Coordinate Ascent** (SPDCA) method. Let $\boldsymbol{\omega}_k = [\mathbf{v}_c]_{c \in \Gamma_k}$ be the model parameters w.r.t. cliques in Γ_k, where $k = 1, ..., t$.

Proposition 1. *Let* $\mathcal{U}^t = \mathcal{V} \cup \mathcal{C}^t$, $\mathbf{w} = [\mathbf{u}; \boldsymbol{\omega}_1^\top; ...; \boldsymbol{\omega}_t]$ *and* $\Phi^{\mathcal{U}^t}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i) = [\ \Phi^{\mathcal{V}}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i); \ \Phi^{\mathcal{C}^t}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i; \boldsymbol{\eta})]$. *Problem (8) is a conic dual form of the following* $\ell^2_{2,1}$*-regularized problem:*

$$\begin{aligned} \min_{\mathbf{v}} \quad & \frac{\lambda}{2}||\mathbf{u}||^2 + \frac{\lambda}{2}(\sum_{k=1}^{t} \|\boldsymbol{\omega}_k\|)^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i, \\ \text{s.t.} \quad & \mathbf{w}^\top \ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i) \geq \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i, \ \forall i, \forall \mathbf{y} \in \mathcal{Y} \backslash \mathbf{y}_i. \end{aligned} \tag{10}$$

Problem (10) is a kind of group lasso problem [24]. The non-smooth $\ell^2_{2,1}$-norm regularizer $\frac{\lambda}{2}(\sum_{k=1}^{t} \|\boldsymbol{\omega}_k\|)^2$ would encourage **sparsity** among $\boldsymbol{\omega}_k$'s [34, 40]. If a clique is indeed irrelevant to the output, it will possibly be dropped out due to the $\ell^2_{2,1}$-norm regularization.

Solving (10) is non-trivial due to the non-smooth objective function. Motivated by [29], we propose solve it by a proximal primal-dual ascent method. To achieve this, we introduce an additional smooth regularizer $\frac{\sigma}{2}\|\boldsymbol{\omega}\|^2$ to the objective of (10), and minimize the following objective instead:

$$f(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{u}\|^2 + \frac{\sigma}{2}\|\boldsymbol{\omega}\|^2 + \frac{\lambda}{2}(\sum_{k=1}^{t} \|\boldsymbol{\omega}_k\|)^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i, \tag{11}$$

where $\xi_i = \max_{\forall \mathbf{y} \in \mathcal{Y} \backslash \mathbf{y}_i} \left(\Delta(\mathbf{y}, \mathbf{y}_i) - \mathbf{w}^\top \ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i)\right)$, $\forall i$.

Proposition 2. *Let* $\mathbf{w}^*$ *be an* $\frac{\epsilon}{2}$*-accurate minimizer of (11). With a sufficiently small* σ, $\mathbf{w}^*$ *is also an* ϵ*-accurate solution of (10).*

Let $\Omega(\mathbf{w}) := \frac{1}{2}\|\mathbf{u}\|^2 + \frac{\sigma}{2\lambda}\|\boldsymbol{\omega}\|^2 + \frac{1}{2}(\sum_{k=1}^{t} \|\boldsymbol{\omega}_k\|)^2$, where $\mathbf{w} = [\mathbf{u}; \boldsymbol{\omega}]$. Let $L_i(\mathbf{w}^\top \ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i)) := \xi_i$. Then, the problem can be transformed into the following problem

$$\min_{\mathbf{w}} \ \lambda\Omega(\mathbf{w}) + \frac{1}{n}\sum_{i=1}^{n} L_i(\mathbf{w}^\top \ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i)). \tag{12}$$

Now, $\Omega(\mathbf{w})$ is 1-strongly convex and $L_i(\mathbf{w}^\top \ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i))$ is γ-Lipschitz for some γ [29]. Let $\nabla\Omega^*(\mathbf{z}) = \arg\max_{\mathbf{w}} \mathbf{w}^\top \mathbf{z} - \Omega(\mathbf{w})$ be the conjugate of $\Omega(\mathbf{w})$,[1] and L_i^* be the conjugate of L_i. The conjugate dual of problem (12) can be written as

$$\max_{\boldsymbol{\alpha}} \ D(\boldsymbol{\alpha}), \ \ \text{s.t.} \quad \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \leq 1, \forall i \in [n],$$

[1] The computation of $\nabla\Omega^*(\mathbf{z})$ is put in the supplementary file.

where $D(\boldsymbol{\alpha}) = -\lambda\Omega^* \left(\frac{1}{\lambda n}\sum_{i=1}^{n}\sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}}\ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)\right) - \frac{1}{n}\sum_{i=1}^{n} L_i^*(-\boldsymbol{\alpha}_{i\mathbf{y}})$, $\boldsymbol{\alpha}_{i\mathbf{y}} = [\alpha_{i\mathbf{y}}]_{\mathbf{y}\neq\mathbf{y}_i}$ and $L_i^*(\boldsymbol{\alpha}_{i\mathbf{y}}) = \sum_{\mathbf{y}\neq\mathbf{y}_i} \triangle(\mathbf{y},\mathbf{y}_i)\alpha_{i\mathbf{y}}$.

The basic idea of SPDCA is to increase the conjugate dual objective of (12) w.r.t. a single variable as much as possible. Specifically, for a randomly chosen instance i, we find an active label $\mathbf{y}^*$ by solving

$$\mathbf{y}^* = \arg\max_{\mathbf{y}\neq\mathbf{y}_i} \Delta(\mathbf{y},\mathbf{y}_i) - \mathbf{w}^\top\ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i). \tag{13}$$

Given $\mathbf{y}$,[2] we need to update $\mathbf{w}$ and dual component $\alpha_{i\mathbf{y}}$ by computing $\delta_{i\mathbf{y}}$ to maximize the following dual objective:[3]

$$-\lambda\Omega^*\left(\mathbf{w}^{(h-1)} + \frac{1}{\lambda n}\delta_{i\mathbf{y}}\ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)\right) - \frac{1}{n}L_i^*(-(\alpha_{i\mathbf{y}} + \delta_{i\mathbf{y}})).$$

Let $a = ||\ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)||_2$ and $d = \mathbf{w}^{h-1\top}\ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)$. It is equivalent to compute $\delta_{i\mathbf{y}}$ to maximize a proximal objective

$$\max_{\delta_{i\mathbf{y}}} \quad (\triangle(\mathbf{y},\mathbf{y}_i) - d)\,\delta_{i\mathbf{y}} - \frac{a^2}{2\lambda n}\delta^2_{i\mathbf{y}}.$$

Regarding this problem, we have a closed-form solution $\delta_{i\mathbf{y}} = \frac{\lambda n(\triangle(\mathbf{y},\mathbf{y}_i)-d)}{a^2}$. To deal with the issue when $||\ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)||_2 = 0$, we update $\delta_{i\mathbf{y}} = \frac{\lambda n(\triangle(\mathbf{y},\mathbf{y}_i)-d)}{\theta(a^2+\nu)}$, where $\theta \geq 1$ and $\nu > 0$. The detailed algorithm is shown in Algorithm 2.

Algorithm 2 Stochastic Proximal Dual Coordinate Ascent.

```
Given parameter $\lambda$, $\nu$, $\theta > 1$ and $\epsilon$. Initialize $\mathbf{w} = [\mathbf{u}; \boldsymbol{\omega}_1; ..., \boldsymbol{\omega}_{t-1}; \mathbf{0}]$ (For warm start).
For $h = 1, ..., H$.
  Let $f_h = 0$.
  For $z = 1, ..., n$.
    Randomly pick $i \in \{1, ..., n\}$.
    Find $\mathbf{y} = \arg\max_{\mathbf{y}\neq\mathbf{y}_i} \Delta(\mathbf{y},\mathbf{y}_i) - \mathbf{w}^\top\ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)$.
    Compute $\xi_i = \Delta(\mathbf{y}^*,\mathbf{y}_i) - \mathbf{w}^\top\ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)$.
    Update $\delta_{i\mathbf{y}} = \frac{\xi_i}{\theta(\frac{1}{\lambda n}||\boldsymbol{\Psi}(\mathbf{x}_i,\mathbf{y})||_2^2+\nu)}$.
    Let $\mathbf{z}_i = \mathbf{w}_{i-1} + \delta_{i\mathbf{y}}\ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)$, and $f_h := f_h + \xi_i$.
    Update $\mathbf{w}_i = \Omega^*(\mathbf{z}_i)$.
  end
  Let $f_h := f_h/n + \lambda\Omega(\mathbf{w})$.
  If ($h > 2$ and $f_h > f_{h-1}$), let $\theta := 2\theta$.
  If ($h \geq 5$ and $\frac{|f_h - f_{h-5}|}{f_{h-5}} \leq \epsilon$), quit and return $\delta_{i\mathbf{y}}$.
end
```

Theorem 2. *It is sufficient for Algorithm (2) to obtain an expected ϵ-duality gap after $O(\frac{R^2}{\lambda\epsilon})$ iterations, where R is an upper bound on $||\ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)||_2$.*

Algorithm 2 performs in a stochastic manner, and does not explicitly maintain $\boldsymbol{\alpha}_{i\mathbf{y}}$ which is required in our active clique section. Note that if $\mathbf{w}^*$ is an ϵ-accurate solution, then $\boldsymbol{\alpha}$ is an ϵ-accurate solution. For any active $\mathbf{y}$ of the last epoch, its initial value is $\mathbf{0}$. Then the update rule for $\boldsymbol{\alpha}_{iy}$ becomes $\boldsymbol{\alpha}_{iy} := \delta_{i\mathbf{y}}$. Therefore, we directly choose $\delta_{i\mathbf{y}}$ of the last epoch as $\boldsymbol{\alpha}_{iy}$.

4.5. Joint Feature Mapping Selection

The joint feature mapping $\boldsymbol{\Psi}(\mathbf{y},\mathbf{x})$ plays an essential role. Let $\boldsymbol{\psi}(\mathbf{x})$ denote the basic input feature. Following [31], we consider the joint feature mapping below

$$\boldsymbol{\Psi}(\mathbf{y},\mathbf{x}) = \boldsymbol{\Upsilon}(\mathbf{y}) \otimes \boldsymbol{\psi}(\mathbf{x}),$$

where $\otimes$ denotes the tensor product, and $\boldsymbol{\Upsilon}(\mathbf{y}) = [\Upsilon_c(\mathbf{y}^c)]_{c\in\mathcal{C}\cup\mathcal{V}}$ consisting of labeling indicators $\Upsilon_c(\mathbf{y}^c) = [\mathbf{1}_{\{\mathbf{y}^c=\mathbf{u}^c\}}]_{\mathbf{u}^c\in\mathcal{Y}_c}$. In this way, one can easily capture the dependency between labels without the need for prior alignment of input and output features [31]. Other joint feature mappings can be applied here too.

4.6. Efficient Inference on Sparse Graphs

In both training and prediction, we need to predict $\mathbf{y}^*$ by solving a standard MAP inference problem:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}\neq\mathbf{y}_i} \quad \tau\Delta(\mathbf{y},\mathbf{y}_i) - \mathbf{w}^\top\ \Phi^{\mathcal{U}}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i), \tag{14}$$

where $\mathcal{U} = \mathcal{V}\cup\mathcal{C}$ denote the clique set, and $\tau = 0$ is for prediction and $\tau = 1$ for training. Note that our learnt graph may have loops. In this paper, we adopt the Max-Product Linear Programming [14] to complete the inference. This method is guaranteed to converge, and has optimal certificate when the gap between the dual and primal vanishes, regardless of loops. In our setting, it finds an exact solution in most cases since the graph is sparse [45].

5. Experiments

We evaluate the proposed Clique Generating Machine (CGM) on one synthetic data set and three multi-label image annotation data sets, namely the nature **Scene** data set [4], **PASCAL VOC 2007** data set (denoted by PASCAL07),[5] and **PASCAL VOC 2012** data set (denoted by PASCAL12 [9]. The **Scene** data set contains 2407 images with 6 labels, and each image is represented by 297 features. The PASCAL07 data set contains 9963 images with 20 labels. Here, we adopt two kinds of features to represent PASCAL2007, i.e., the dense SIFT features (denoted by PASCAL2007-DSIFT) of 3000 dimensions, and deep CNN (convolutional neural network) features [19]. For **PASCAL VOC 2012**,

[2] For convenience, we drop the superscript $*$ from $\mathbf{y}^*$.

[3] The computation of $\Omega^*(\mathbf{z})$ can be found in supplementary file.

[4] http://mulan.sourceforge.net/datasets-mlc.html.

[5] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/.

Table 2. Hamming loss on toy testing data set.

Method	CGM	NO-Edge	FULL-Edge	CL-TREE	HLLM-L1 [28]	LSG21 [4]	BR [23]	PLST [32]	MMOC [44]	LEAD [43]	PLEM [22]
HammingLoss	**0.011**	0.053	0.028	0.044	0.024	0.059	0.054	0.055	0.064	0.0484	0.064

we use CNN features only. In our experiment, we extract CNN features via Caffe tool box,[6] which represent each image by 4096 features. The statistics of these data sets are summarized in Table 1.

Table 1. Data sets summary

Data set	#instances	#features	#labels
Scene	2407	297	6
PASCAL07-DSift	9963	3000	20
PASCAL07-CNN	9963	4096	20
PASCAL12-CNN	11540	4096	20

5.1. Baseline Methods and Performance Measures

We compare CGM with following baselines: BR (Binary Relevance) [23]; LEAD (Multi-label learning by exploiting label dependency) [43]; PLST (Principal label space transformation) [32]; MMOC(Maximum margin output coding) [44]; PLEM (Probabilistic enhancement model) [22]. HLLM-L1 (Hierarchical log linear model with overlapping group ℓ_1-regularization) [28]; SSVM-CT (Structured SVM with ChowLiu Tree). BR is considered as a basic baseline; while PLEM, HLLM-L1, SSVM-CT and MMOC are considered as state-of-the-arts. We adopt LIBlinear SVM to build binary classifiers for BR [10], which is the same as in [4]. We implement PLEM ourselves; while the sources of other methods are publicly available.

We adopt four widely used performance criteria as the comparison metrics, i.e., Hamming Loss, Macro-F1, Micro-F1, Micro-Recall [44, 4, 22]. All the measures are computed through the codes provided in [44]. The mean average precisions (mAP) is classical metric for measuring the performance of rankings [42]. However, in our methods and PLEM [22], the labels are directly predicted by solving the MAP inference problem in (14) (each predicted label is either 0 or 1), and the ranking information is not available. Therefore, we do not adopt mAP measure. On each data set, we compare all the methods by 5-cross validation. The mean and deviation of each criterion are recorded.

All the experiments are conducted in Matlab on a PC installed a 64-bit operating system with an Intel(R) Core(TM) i7-4770 CPU (3.40GHz with single-thread mode) and 8GB memory.

5.2. Experimental Results on Synthetic Data Set

We first conduct a synthetic experiment to demonstrate the performance of the proposed method. We generate a synthetic data set of 10 labels (e.g., $\mathcal{V} = \{1, ..., 10\}$) and a basic input feature vector $\boldsymbol{\psi}(\mathbf{x}) \in \mathbb{R}^{10}$, where each dimension is sampled from a Gaussian distribution $\mathcal{N}(0, 1)$. We make 4 edges $\widehat{\mathcal{C}} = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}\}$ as the ground-truth relevant edges. We then sample 2000 instances for training and 1000 instances for testing. For each node (*i.e.* a label here), we generate a Gaussian random weight vector $\mathbf{u} \in \mathbb{R}^5$. Since $y^i \in \{0, 1\}$, each edge has 4 possible states, namely 00, 01, 10, and 11 [28]. For each states, we generate a Gaussian random vector $\mathbf{v} \in \mathbb{R}^5$ as the ground-truth edge parameters. After generating the ground-truth parameter denoted as $\mathbf{w}$ (constructed from $\mathbf{u}$ and $\mathbf{v}$), we generate the ground-truth labels $\widehat{\mathbf{y}}$ by solving $\widehat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}} \mathbf{w}^\top \boldsymbol{\Psi}(\mathbf{y}, \mathbf{x})$. We set $\lambda = 0.0001$ and $B = 2$ for CGM.

In the synthetic experiments we investigate: 1) whether the CGM can successfully find the ground-truth edges, and 2) whether these edges are helpful to improve the prediction performance. To do so, we adopt the following baselines for comparison: SSVM with all edges (denoted by FULL-Edge), SSVM without any edges (denoted by NO-Edge), and SSVM with ChowLiu Tree based graphs (denoted by CL-TREE). Note that above SSVMs based methods have the same primal objective function with ours (once the graph is given), thus we can plot their primal objective values versus number of epochs in Figure 2.

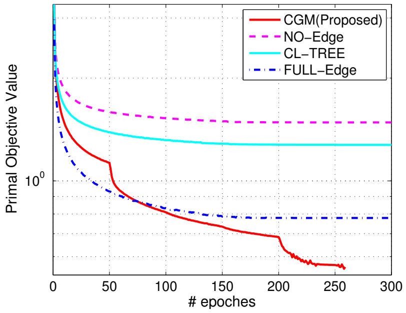


Figure 2. Primal objective values versus # epochs

From Figure 2, CGM achieves the fastest convergence speed and the lowest objective value. In our experiment, it successfully finds the ground-truth edges; while the ChowLiu Tree cannot identify any of the ground-truth edges. In fact, SSVM without edges and SSVM with ChowLiu Tree graphs cannot decrease the objective significantly after 100 epochs. SSVM with all edges converges fast at the beginning epochs, but it become very slow after 150 epochs.

Note that the labels of this synthetic data set is very dense. Therefore, the Hamming loss is a sufficient criterion

[6] http://caffe.berkeleyvision.org/.

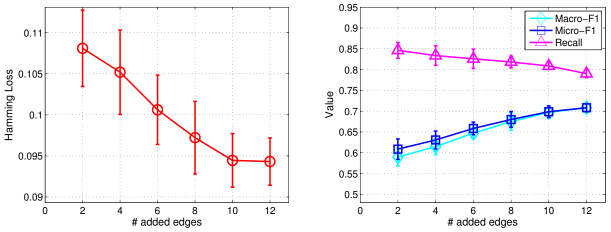
for comparison. We reported the Hamming loss on testing data set in Table 2. From this table, CGM indeed achieves the lowest testing Hamming loss compared to other methods. HLLM-L1, which imposes sparsity on edges, shows better results than other methods except CGM. However, it has much higher memory requirements than CGM, thus it is not scalable to large-scale problems.

5.3. Experimental Results on Real-world Data Sets

In the experiment on real-world data sets, we first report the mean and deviation of the four criteria obtained by various methods. The results are reported in Table 3. From Table 3, we draw the following conclusions.

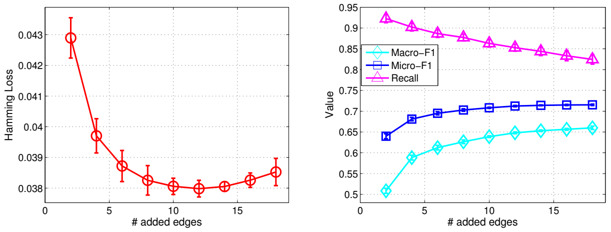
- On all data sets, CGM shows better or comparable performance. In particular, CGM shows the best performance on PASCAL07 and PASCAL2012 using CNN features in terms of all four measures. Moreover, it also achieves the best results on all data sets in terms of Hamming loss and Macro-F1.
- From Table 3, CGM shows significant improvements over BR in terms all performance measures. This demonstrates that, exploiting the label dependency indeed improves the classification performance.
- CGM shows significantly better performance than SSVM-CT in terms of most measures, which demonstrates the effectiveness of CGM. In Figure 1 of the **Introduction** section, we have compared the graphs obtained by ChowLiu Tree and CGM, and the graph obtained by CGM is more reasonable.
- CGM shows better performance than PLEM [22], a method relies on a graph learned on labels only, and the features are not adopted to adjust the graph. Similarly, the LEAD method is based on a pre-learned DAG graph. Therefore, its performance is also limited.
- The HLLM-L1 method, which enforces sparsity on edges, shows comparable performance to CGM on Scene data set. But its results on PASCAL07 and PASCAL2012 data sets are absent due to the out-of-memory issue.
- The PLST method shows relatively worse performance than others, but it is very efficient on large-scale problems of many labels. The MMOC method show superior results over BR and PLST, but the training of output codes is time consuming.

We further show the performance variation of CGM versus the number of edge cliques on Scene and PASCAL07-CNN data sets. The results are shown in Figures 3 and 4. We draw the following conclusions. Firstly, on the two data sets, adding the relevant label dependencies improves the performance measures, such as Hamming loss and F1 measures. However, adding too many label dependencies does not necessarily improve the performance. For example, on Scene data, the performance of CGM does not show significant improvement when there are more than 10 edges. On PASCAL07 data, the performance of CGM in terms of Hamming loss indeed degrades when there are more than 12 edges. In other words, adding too many edges (cliques) (especially irrelevant edges (cliques)) may even degrade the prediction performance.



(a) Hamming loss (b) Other measures

Figure 3. Performance variations v.s. # edges on Scene



(a) Hamming loss (b) Other measures

Figure 4. Performance variations v.s. # edges on PASCAL07

5.4. Further Comparison

In Table 4, we compare our results on Scene and PASCAL07 with four additional latest methods including LSG21 [4], MLF [18], MLLS [41], and MLRF [18] using the published results in [4]. On Scene data set, we use the same features. However, the PASCAL07 data used in [4] are represented by GIST features [25]. Therefore, the comparison on PASCAL07 might not be as indicative as we would expect. In fact, during the preparation of this paper, we have attempted to conduct experiments with the compared methods on PASCAL07 with GIST features, but the experimental results do not match the published results. This might be produced by data pre-processing, which, however, is unclear to us. Maybe due to the same reason, though we used the LSG21's source codes, we did not manage to reproduce their published results. Nevertheless, CGM outperforms comparison methods on Scene data in terms of F1 measures. On PASCAL07, CGM with CNN features achieves significantly better results than others. Interestingly, this comparison also demonstrates the superiority of CNN local features on multi-label image classification tasks.

Table 3. Performance comparison via 5-fold cross validations

Measure	Methods	Data sets			
		Scene	PASCAL07-DSift	PASCAL07-CNN	PASCAL12-CNN
Hamming Loss	BR [23]	0.110±0.003	0.081±0.001	0.052±0.001	0.052±0.001
	LEAD [43]	0.100±0.002	0.071±0.001	0.045±0.001	0.046±0.001
	PLST [32]	0.118±0.003	0.091±0.001	0.054±0.001	0.060±0.001
	MMOC [44]	0.098±0.009	0.076±0.001	0.045±0.001	0.047±0.001
	PLEM [22]	0.099±0.003	0.080±0.001	0.049±0.003	0.046±0.001
	HLLM-L1 [28]	0.104±0.002	–	–	–
	SSVM-CT	0.100±0.002	0.068±0.001	0.041±0.001	0.043±0.001
	CGM	**0.094±0.003**	**0.065±0.001**	**0.038±0.001**	**0.040±0.001**
Macro-F1	BR [23]	0.678±0.009	0.279±0.004	0.623±0.006	0.424±0.014
	LEAD [43]	0.679±0.009	0.0.170±0.003	0.595±0.009	0.555±0.009
	PLST [32]	0.608±0.006	0.080±0.002	0.502±0.008	0.160±0.003
	MMOC [44]	0.696±0.020	0.184±0.005	0.577±0.003	**0.629±0.005**
	PLEM [22]	0.693±0.006	0.276±0.003	0.630±0.003	0.605±0.007
	HLLM-L1 [28]	0.687±0.005	–	–	–
	SSVM-CT	0.610±0.009	0.214±0.008	0.634±0.005	**0.629±0.004**
	CGM	**0.708±0.010**	**0.348±0.008**	**0.648±0.002**	0.623±0.005
Micro-F1	BR [23]	0.676±0.010	0.384±0.003	0.657±0.004	0.389±0.013
	LEAD [43]	0.682±0.010	0.315±0.004	0.659±0.006	0.619±0.011
	PLST [32]	0.608±0.007	0.285±0.005	0.617±0.009	0.436±0.006
	MMOC [44]	**0.719±0.017**	0.364±0.004	0.682±0.002	0.666±0.003
	PLEM [22]	0.691±0.010	0.378±0.003	0.673±0.003	0.639±0.008
	HLLM-L1 [28]	0.698±0.014	–	–	–
	SSVM-CT	0.682±0.010	0.384±0.005	0.701±0.006	0.672±0.007
	CGM	0.703±0.010	**0.434±0.005**	**0.712±0.002**	**0.681±0.007**
Recall	BR [23]	0.706±0.008	0.438±0.007	0.669±0.010	0.267+0.009
	LEAD [43]	0.785±0.008	0.407±0.004	0.791±0.012	0.780±0.006
	PLST [32]	0.740±0.013	0.334±0.003	0.693±0.023	0.069±0.006
	MMOC [44]	0.745±0.017	**0.558±0.003**	0.825±0.007	0.702±0.013
	PLEM [22]	0.773±0.007	0.438±0.004	0.688±0.009	0.666±0.005
	HLLM-L1 [28]	0.789±0.010	–	–	–
	SSVM-CT	0.785±0.008	0.478±0.005	0.810±0.008	0.742±0.001
	CGM	**0.790±0.010**	0.426±0.012	**0.853±0.001**	**0.819±0.015**

[1] The results of HLLM-L1 [28] on PASCAL data sets are absent due to out-of-memory issue.

Table 4. Performance comparison on Scene and PASCAL07

Measure	Methods	Data sets	
		Scene	PASCAL07
Macro-F1	LSG21 [4]	0.675	0.407
	MLF [18]	0.684	0.332
	MLLS [41]	0.472	0.322
	MLRF [18]	0.682	0.326
	CGM	**0.708**	**0.648**(CNN)
Micro-F1	LSG21 [4]	0.682	0.307
	MLF [18]	0.690	0.221
	MLLS [41]	0.472	0.215
	MLRF [18]	0.686	0.214
	CGM	**0.703**	**0.712**(CNN)
Precision	LSG21 [4]	**0.706**	0.309
	MLF [18]	0.687	0.221
	MLLS [41]	0.473	0.215
	MLRF [18]	0.705	0.22
	CGM	0.640	**0.604**(CNN)

6. Conclusion

A clique generating machine (CGM) has been proposed to learn graph structures for multi-label image classification. We propose a convex programming model (which has exponentially many constraints) to learn sparse graph structures. To solve this model, a cutting plane algorithm is applied, which iteratively activates a group of cliques. An efficient stochastic proximal dual coordinate method has also been proposed to solve the non-smooth subproblems. Experimental results on a synthetic problem demonstrate that CGM indeed can identify the ground-truth cliques from structured labels. On three multi-label image data sets, the proposed CGM has shown much improved results compared to existing methods in terms of several performance measures.

The source code and relevant data sets are available at `http://www.tanmingkui.com/cgm.html`.

References

[1] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recogn.*, 37(9):1757–1771, 2004.

[2] J. K. Bradley and C. Guestrin. Learning tree conditional random fields. In *Proc. Int. Conf. Mach. Learn.*, 2010.

[3] S. S. Bucak, P. Kumar Mallapragada, R. Jin, and A. K. Jain. Efficient multi-label ranking for multi-class learning: application to object recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009.

[4] X. Cai, F. Nie, W. Cai, and H. Huang. New graph structured sparsity model for multi-label image annotations. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2013.

[5] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inform. Theory*, 14(3):462–467, 1968.

[6] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On label dependence and loss mini-

mization in multi-label classification. *Mach. Learn.*, 88(1-2):5–45, 2012.

[7] K. Dembczynski, W. Waegeman, and E. Hüllermeier. An analysis of chaining in multi-label classification. In *Proc. Eur. Conf. Artificial Intell.*, 2012.

[8] S. Ding, G. Wahba, and X. Zhu. Learning higher-order graph structure with features by structure penalty. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 253–261, 2011.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.

[10] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.*, pages 1871–1874, 2008.

[11] T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *Proc. Int. Conf. Mach. Learn.*, 2008.

[12] J. Fürnkranz, E. Hüllermeier, E. L. Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Mach. Learn.*, 73(2):133–153, 2008.

[13] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proc. ACM Int. Conf. Info. & Knowledge Management*, pages 195–200. ACM, 2005.

[14] A. Globerson and T. S. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Proc. Adv. Neural Inf. Process. Syst.*, 2008.

[15] Y. Guo and S. Gu. Multi-label classification using conditional dependency networks. In *Proc. Int. Joint Conf. Artificial Intell.*, 2011.

[16] D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *Proc. Adv. Neural Inf. Process. Syst.*, volume 22, pages 772–780, 2009.

[17] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[18] D. Kong, C. Ding, and H. H. H. Zhao. Multi-label relieff and f-statistic feature selections for image annotation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 1097–1105, 2012.

[20] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *ICML*, 2013.

[21] J. D. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *Proc. Int. Conf. Mach. Learn.*, 2001.

[22] X. Li, F. Zhao, and Y. Guo. Multi-label image classification with a probabilistic label enhancement model. In *Proc. Uncertainty in Artificial Intell.*, 2014.

[23] O. Luaces, J. Díez, J. Barranquero, J. J. del Coz, and A. Bahamonde. Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, 1(4):303–313, 2012.

[24] A. F. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. Online multiple kernel learning for structured prediction. Technical report, 2010.

[25] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, 155:23–36, 2006.

[26] S. Paisitkriangkrai, C. Shen, Q. Shi, and A. van den Hengel. RandomBoost: Simplified multiclass boosting through randomization. *IEEE Trans. Neural Netw. Learn. Syst.*, 25(4):764–779, 2014.

[27] S. Paisitkriangkrai, C. Shen, and A. van den Hengel. A scalable stagewise approach to large-margin multiclass loss-based boosting. *IEEE Trans. Neural Netw. Learn. Syst.*, 25(5):1002–1013, 2014.

[28] M. W. Schmidt and K. P. Murphy. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proc. Int. Conf. Artificial Intell. & Statistics*, pages 709–716, 2010.

[29] S. Shalev-Shwartz and T. Zhang. Proximal stochastic dual coordinate ascent, 2012.

[30] M. Sion. On general minimax theorems. *Pacific J. Math*, 8(1):171–176, 1958.

[31] H. Su and J. Rousu. Multilabel classification through random graph ensembles. *Mach. Learn.*, 2014.

[32] F. Tai and H.-T. Lin. Multilabel classification with principal label space transformation. *Neural Comput.*, 24(9):2508–2542, 2012.

[33] M. Tan, I. Tsang, and L. Wang. Towards ultrahigh dimensional feature selection for big data. *J. Mach. Learn. Res.*, 15:1371–1429, 2014.

[34] M. Tan, I. W. Tsang, L. Wang, and X. Zhang. Convex matching pursuit for large-scale sparse coding and subset selection. In *Proc. AAAI*, 2012.

[35] M. Tan, L. Wang, and I. Tsang. Learning sparse svm for feature selection on very high dimensional datasets. In *Proc. Int. Conf. Mach. Learn.*, pages 1047–1054, 2010.

[36] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. In *J. Mach. Learn. Res.*, pages 1453–1484, 2005.

[37] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2010.

[38] G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Proc. Euro. Conf. Mach. Learn.*, pages 406–417. Springer, 2007.

[39] Q. Wu, Y. Ye, H. Zhang, T. W. S. Chow, and S. S. Ho. ML-TREE: A tree-structure-based approach to multilabel learning. *IEEE Trans. Neural Netw. Learn. Syst.*, 2014.

[40] S. Xiao, W. Li, D. Xu, and D. T. 2015. Accelerating low rank representation via reformulation with factorized data. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[41] K. Yu, S. Yu, and V. Tresp. Multi-label informed latent semantic indexing. In *Proc. Annual ACM SIGIR Conf.*, pages 258–265. ACM, 2005.

[42] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proc. Annual ACM SIGIR Conf.*, 2007.

[43] M.-L. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *Proc. ACM Int. Conf. Knowledge Discovery & Data Mining*, 2010.

[44] Y. Zhang and J. Schneider. Maximum margin output coding. In *Proc. Int. Conf. Mach. Learn.*, 2012.

[45] Z. Zhang, Q. Shi, Y. Zhang, C. Shen, and A. van den Hengel. Constraint reduction using marginal polytope diagrams for MAP LP relaxations, 2013.

Supplementary Materials for Paper 1915

Due to the page limit, in the main paper, we omitted some details of Algorithms 1 and 2, and all the proof of the main conclusions. In this supplementary file, we first present the omitted details of Algorithms 1 and 2, and then present the derivations of the main conclusions in the paper.

A. More Details about Algorithm 1 and Algorithm 2

For convenience, we first present the stopping conditions of Algorithm 2.

A.1. Stopping Condition of Algorithm 2

It is usually non-trivial to set a proper stopping condition for stochastic optimization algorithms. Usually, we can stop an algorithm when the objective value does not change significantly. For example, we can stop Algorithm 2 if the primal objective value cannot decrease significantly. Unfortunately, computing the primal objective value f_h is very expensive. Moreover, **the primal objective value does not monotonically decrease w.r.t.** h. Therefore, we propose to stop Algorithm 2 if $h > 5$ and

$$\frac{|f_h - f_{h-5}|}{f_{h-5}} \leq \epsilon.$$

Here, f_h is computed as in Algorithm 2, and it approximates the primal objective value of the SSVM subproblem. In our implementation, we choose and fix $\epsilon = 0.005$.

A.2. Inequality Constraint Handling in Subproblem Optimization

Note that the conjugate dual of the subproblem in (12) is

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\lambda\Omega^*\left(\frac{1}{\lambda n}\sum_{i=1}^{n}\sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}}\mathbf{w}^\top\ \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)\right) - \frac{1}{n}\sum_{i=1}^{n}L_i^*(-\boldsymbol{\alpha}_{i\mathbf{y}}), \\ & \text{s.t.} \sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}} \leq 1, \forall i \in [n]. \end{aligned} \tag{15}$$

where $\boldsymbol{\alpha}_{i\mathbf{y}} = [\alpha_{i\mathbf{y}}]_{\mathbf{y}\neq\mathbf{y}_i}$ and L_i^* denotes the conjugate of the loss function L_i. Note that in (15), we have an **inequality constraint** $\sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}} \leq 1$ on $\boldsymbol{\alpha}_{i\mathbf{y}}$.

In Algorithm 2, we do not store $\alpha_{i\mathbf{y}}$ explicitly, thus we can not handle the inequality constraint directly. When the inequality constraint is ignored, the update rule $\delta_{i\mathbf{y}} = \frac{\lambda n(\triangle(\mathbf{y},\mathbf{y}_i)-d)}{(a^2+\nu)}$ may be too aggressive (e.g. $\delta_{i\mathbf{y}}$ may be too large). To address this, we use a scaled update rule $\delta_{i\mathbf{y}} = \frac{\lambda n(\triangle(\mathbf{y},\mathbf{y}_i)-d)}{\theta(a^2+\nu)}$ instead, where $\theta > 1$. In our implementation, we initialize $\theta = 2$ and update $\theta := 2\theta$ when f_h does not decrease (See Algorithm 2).

A.3. Stopping Condition of Algorithm 1

Similar in Algorithm 2, we stop Algorithm 1 when the primal objective value does not decrease significantly. Let $f^t = f_h$, where f_h is the approximated primal objective value obtained from Algorithm 2. Then, we stop Algorithm 1 if $t > 2$ and

$$\frac{|f^t - f^{t-2}|}{f_{t-2}} \leq \epsilon_o.$$

In our implementation, we choose and fix $\epsilon_o = 0.05$.

B. Lagrangian Dual in (4) of Problem (3)

Proof. Note that $\Phi^{\mathcal{V}}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i) := \mathbf{\Psi}_{\mathcal{V}}(\mathbf{y}_i, \mathbf{x}_i) - \mathbf{\Psi}_{\mathcal{V}}(\mathbf{y}, \mathbf{x}_i)$, $\Phi^{\mathcal{C}}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i; \boldsymbol{\eta}) := \mathbf{\Psi}_{\mathcal{C}}(\mathbf{y}_i, \mathbf{x}_i; \boldsymbol{\eta}) - \mathbf{\Psi}_{\mathcal{C}}(\mathbf{y}, \mathbf{x}_i; \boldsymbol{\eta})$. The Lagrangian function of the inner minimization problem in (3) can be written as:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i - \boldsymbol{\beta}^\top \boldsymbol{\xi} + \sum_{i=1}^{n} \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \left(\Delta(\mathbf{y}, \mathbf{y}_i) - \mathbf{u}^\top \, \Phi^{\mathcal{V}}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i) - \mathbf{v}^\top \, \Phi^{\mathcal{C}}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i; \boldsymbol{\eta}) - \xi_i\right) . \tag{16}$$

Let $\boldsymbol{\alpha} := [\alpha_{1\mathbf{y}}, ..., \alpha_{n\mathbf{y}}]^\top$. The KKT condition of (16) can be written as

$$\frac{\partial \mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \mathbf{u}} = 0 \Rightarrow \mathbf{u} = \frac{1}{\lambda}\sum_{i=1}^{n} \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \, \Phi^{\mathcal{V}}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i); \tag{17}$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \mathbf{v}} = 0 \Rightarrow \mathbf{v} = \frac{1}{\lambda}\sum_{i=1}^{n} \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \, \Phi^{\mathcal{C}}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i; \boldsymbol{\eta}); \tag{18}$$

$$\frac{\partial \mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial \xi_i} = 0 \Rightarrow \frac{1}{n} = \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} + \beta_i; \tag{19}$$

$$\boldsymbol{\alpha} \succeq \mathbf{0}, \text{ and } \boldsymbol{\beta} \succeq \mathbf{0}. \tag{20}$$

Let $\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{R}^l | \boldsymbol{\alpha} \succeq 0, \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \leq \frac{1}{n}\}$ be the domain of $\boldsymbol{\alpha}$. Define

$$\mathbf{u}(\boldsymbol{\alpha}) := \sum_{i=1}^{n} \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \, \Phi^{\mathcal{V}}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i) \text{ and } \mathbf{v}(\boldsymbol{\alpha}, \boldsymbol{\eta}) := \sum_{i=1}^{n} \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \, \Phi^{\mathcal{C}}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i; \boldsymbol{\eta}). \tag{21}$$

Substituting the above relations into (16), the Lagrangian dual of the inner problem of (3) can be written as

$$\max_{\boldsymbol{\alpha} \in \mathcal{A}} \quad -\frac{1}{2\lambda}||\mathbf{u}(\boldsymbol{\alpha})||^2 - \frac{1}{2\lambda}||\mathbf{v}(\boldsymbol{\alpha}, \boldsymbol{\eta})||^2 + \mathbf{b}^\top \boldsymbol{\alpha}. \tag{22}$$

□

C. Proof of Theorem 1

The proof of Theorem 1 can be adapted from the proof of Theorem 2 in [35].

D. Proof of Proposition 1

Proof. Let $\Omega(\boldsymbol{\omega}) = \frac{1}{2}(\sum_{k=1}^{t} \|\boldsymbol{\omega}_k\|)^2$. Define a cone $\mathcal{Q}_r = \{(\mathbf{u}, v) \in \mathbb{R}^{r+1}, \|\mathbf{u}\|_2 \leq v\}$. Let $z_k = \|\boldsymbol{\omega}_k\|$, we have $\Omega(\mathbf{v}) = \frac{1}{2}(\sum_{k=1}^{t} \|\boldsymbol{\omega}_k\|)^2 = \frac{1}{2}z^2$, where $z = \sum_{k=1}^{t} z_k$, $z_k \geq 0$ and $z \geq 0$. Then, problem (10) can be transformed to the following problem:

$$\min_{z,\mathbf{u},\mathbf{v}} \ \frac{\lambda}{2}||\mathbf{u}||^2 + \frac{\lambda}{2}z^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i, \quad \text{s.t.} \sum_{k=1}^{t} z_k \leq z, \quad (\boldsymbol{\omega}_k, z_k) \in \mathcal{Q}_r,$$
$$\mathbf{w}^\top \, \Phi^{\mathcal{U}^t}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i) \geq \Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i, \ \ \xi_i \geq 0 \, \forall i, \forall \mathbf{y} \in \mathcal{Y} \backslash \mathbf{y}_i.$$

where $\boldsymbol{\omega} = [\boldsymbol{\omega}'_1, ..., \boldsymbol{\omega}'_t]'$. The Lagrangian function of (23) can be written as:

$$\mathcal{L}(z, \mathbf{v}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \gamma, \boldsymbol{\zeta}, \boldsymbol{\varpi}) = \frac{\lambda}{2}||\mathbf{u}||^2 + \frac{\lambda}{2}z^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i + \gamma(\sum_{k=1}^{t} z_k - z) - \sum_{k=1}^{t}(\boldsymbol{\zeta}'_k \boldsymbol{\omega}_k + \varpi_k z_k)$$
$$- \sum_{i=1}^{n} \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \left(\Delta(\mathbf{y}, \mathbf{y}_i) - \xi_i - \left(\Phi^{\mathcal{V}}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i) + \sum_{k=1}^{t} \boldsymbol{\omega}_k^\top \, \Phi^{\Gamma_k}_{\mathbf{y}_i,\mathbf{y}}(\mathbf{x}_i)\right)\right) - \boldsymbol{\beta}^\top \boldsymbol{\xi},$$

where $\boldsymbol{\alpha}$, γ, $\boldsymbol{\zeta}_k$ and ϖ_k are the Lagrangian dual variables to the corresponding constraints. The KKT condition can be expressed as

$$\begin{array}{ll}
\nabla_z \mathcal{L} = \lambda z - \gamma = 0 & \Rightarrow z = \frac{\gamma}{\lambda}; \\
\nabla_{z_k} \mathcal{L} = \gamma - \varpi_k = 0 & \Rightarrow \varpi_k = \gamma; \\
\nabla_{\mathbf{u}} \mathcal{L} = \lambda \mathbf{u} + \sum_{i=1}^n \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \; \Phi^{\mathcal{V}}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i) & \Rightarrow \mathbf{u} = -\frac{1}{\lambda} \sum_{i=1}^n \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \; \Phi^{\mathcal{V}}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i) \\
\nabla_{\boldsymbol{\omega}_k} \mathcal{L} = -\sum_{i=1}^n \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \left(\; \Phi^{\Gamma_k}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i) \right) - \boldsymbol{\zeta}_k = 0 & \Rightarrow \boldsymbol{\zeta}_k = -\sum_{i=1}^n \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \left(\; \Phi^{\Gamma_k}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i) \right); \\
\nabla_{\xi_i} \mathcal{L} = \frac{1}{n} - \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} - \beta_i = 0 & \Rightarrow \frac{1}{n} = \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} + \beta_i; \\
\|\boldsymbol{\zeta}_k\| \leq \varpi_k & \Rightarrow \|\boldsymbol{\zeta}_k\| \leq \gamma; \\
\beta_i \geq 0 & \Rightarrow \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \leq \frac{1}{n}.
\end{array}$$

By substituting the above equations into the Lagrangian function, we have

$$\mathcal{L}(z, \mathbf{v}, \boldsymbol{\xi}, b, \boldsymbol{\alpha}, \gamma, \boldsymbol{\zeta}, \boldsymbol{\varpi}) = -\frac{1}{2\lambda}\gamma^2 - \frac{1}{2\lambda}||\omega(\boldsymbol{\alpha})||^2 + \sum_{i=1}^n \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y}_i).$$

Hence the dual problem of the $\ell^2_{2,1}$-regularized problem can be written as:

$$\begin{array}{ll}
\max\limits_{\gamma, \boldsymbol{\alpha}} & -\frac{1}{2\lambda}\gamma^2 - \frac{1}{2\lambda}||\mathbf{u}(\boldsymbol{\alpha})||^2 + \sum\limits_{i=1}^n \sum\limits_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y}_i) \\
\text{s.t.} & \left\| \sum\limits_{i=1}^n \sum\limits_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \; \Phi^{\Gamma_k}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i) \right\| \leq \gamma, \;\; k = 1, \cdots, t, \\
 & \alpha_i \geq 0, \sum\limits_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \leq \dfrac{1}{n}, \;\; i = 1, \cdots, n.
\end{array}$$

Let $\theta := -\frac{1}{2\lambda}\gamma^2 - \frac{1}{2\lambda}||\mathbf{u}(\boldsymbol{\alpha})||^2 + \sum_{i=1}^n \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y}_i)$, $\boldsymbol{\omega}_k(\boldsymbol{\alpha}, \boldsymbol{\eta}_k) := \sum_{i=1}^n \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \; \Phi^{\Gamma_k}_{\mathbf{y}_i, \mathbf{y}}(\mathbf{x}_i)$ and $g(\boldsymbol{\alpha}, \boldsymbol{\eta}_k) = -\frac{1}{2\lambda}\|\boldsymbol{\omega}_k(\boldsymbol{\alpha}, \boldsymbol{\eta}_k)\|^2 - \frac{1}{2\lambda}||\mathbf{u}(\boldsymbol{\alpha})||^2 + \sum_{i=1}^n \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y}_i)$. We have

$$\begin{array}{ll}
\max\limits_{\theta, \boldsymbol{\alpha}} & \theta, \\
\text{s.t.} & \theta \leq g(\boldsymbol{\alpha}, \boldsymbol{\eta}_k), \;\; k = 1, \cdots, t, \\
 & \alpha_i \geq 0, \; i = 1, \cdots, n.
\end{array}$$

which indeed is in the form of problem (8) by letting $\mathcal{A}$ be the domain of $\boldsymbol{\alpha}$. This completes the proof and brings the connection between the primal and dual formulation. □

E. Computation of $\Omega^*(\mathbf{z})$

The conjugate of $\Omega(\mathbf{w})$ is defined as

$$\Omega^*(\mathbf{z}) = \max_{\mathbf{u}, \boldsymbol{\omega}} \;\; \mathbf{w}^\top \mathbf{z} \; - \left(\frac{1}{2}\|\mathbf{u}\|^2 + \frac{\sigma}{2\lambda}\|\boldsymbol{\omega}\|^2 + \frac{1}{2}(\sum_{k=1}^t \|\boldsymbol{\omega}_k\|)^2 \right).$$

Let $\mathbf{z} = [\mathbf{z}_u; \mathbf{z}_v]$, where $\mathbf{z}_u$ and $\mathbf{z}_v$ are vectors corresponding to $\mathbf{u}$ and $\boldsymbol{\omega}$, respectively. Let $\Upsilon(\boldsymbol{\omega}) = \left(\frac{\sigma}{2\lambda}\|\boldsymbol{\omega} - \frac{\lambda\mathbf{z}_v}{\sigma}\|^2 + \frac{1}{2}(\sum_{k=1}^t \|\boldsymbol{\omega}_k\|)^2\right)$. The optimal solution, i.e. $\nabla\Omega^*(\mathbf{z})$, can be computed by

$$\begin{aligned}
\nabla\Omega^*(\mathbf{z}) &= \arg\max_{\mathbf{u},\boldsymbol{\omega}} \ \mathbf{u}^\top\mathbf{z}_u + \boldsymbol{\omega}^\top\mathbf{z}_v - \left(\frac{1}{2}\|\mathbf{u}\|^2 + \frac{\sigma}{2\lambda}\|\boldsymbol{\omega}\|^2 + \frac{1}{2}(\sum_{k=1}^t \|\boldsymbol{\omega}_k\|)^2\right) \\
&= \left[\arg\min_{\mathbf{u}} \ \left(\frac{1}{2}\|\mathbf{u}\|^2 - \mathbf{u}^\top\mathbf{z}_u\right); \arg\min_{\boldsymbol{\omega}} \Upsilon(\boldsymbol{\omega})\right] \\
&= \left[\mathbf{z}_u; \arg\min_{\boldsymbol{\omega}} \Upsilon(\boldsymbol{\omega})\right],
\end{aligned}$$

In other words, we just need to solve the following problem

$$\min_{\boldsymbol{\omega}} \quad \frac{\sigma}{2\lambda}\|\boldsymbol{\omega} - \frac{\lambda\mathbf{z}_v}{\sigma}\|^2 + \frac{1}{2}\left(\sum_{k=1}^t \|\boldsymbol{\omega}_k\|\right)^2. \tag{23}$$

This is a strictly convex problem, and a unique minimizer can be computed in closed-form [24].

Proposition 3. *Let $\widehat{\boldsymbol{\omega}}$ be an optimal solution of problem (23). Then, $\widehat{\boldsymbol{\omega}}$ is unique, and can be cheaply calculated by Algorithm 3.*

Proof. Please refer the proof in Appendix F of [24]. □

Algorithm 3 Computation of $\Omega^*(\mathbf{z})$.

```
Given $\mathbf{z} = [\mathbf{z}_u; \mathbf{z}_v]$, parameter $s = \frac{\lambda}{\sigma}$ and scalar $T$. Let $\boldsymbol{\omega} = \frac{\lambda\mathbf{z}_v}{\sigma}$.
1: Calculate $\widehat{o}_k = \|\boldsymbol{\omega}_k\|$, where $\boldsymbol{\omega}_k$ is associated with $\boldsymbol{\omega}_k$ for all $k = 1, ..., T$.
2: Sort $\widehat{\mathbf{o}}$ to obtain $\bar{\mathbf{o}}$ such that $\bar{o}_{(1)} \geq ... \geq \bar{o}_{(T)}$.
3: Find $\rho = \max\left\{t \middle| \bar{o}_k - \frac{s}{1+ks}\sum_{i=1}^{k} \bar{o}_i > 0, k = 1, ..., T\right\}$.
4: Calculate a threshold value $\varsigma = \frac{s}{1+\rho s}\sum_{i=1}^{\rho} \bar{o}_i$.
5: Compute $\mathbf{o}$, where $o_k = \begin{cases} \widehat{o}_k - \varsigma, & \text{if } \widehat{o}_k > \varsigma, \\ 0, & \text{Otherwise.} \end{cases}$.
6: Compute $\widehat{\boldsymbol{\omega}}_k = \begin{cases} \frac{o_k}{\|\boldsymbol{\omega}_k\|}\boldsymbol{\omega}_k, & \text{if } o_k > 0, \\ \mathbf{0}, & \text{otherwise,} \end{cases}$
7: Let $\widehat{\boldsymbol{\omega}} = [\boldsymbol{\omega}_k]_{k\in[T]}$. Output $\Omega^*(\mathbf{z}) = [\mathbf{z}_u; \widehat{\boldsymbol{\omega}}]$.
```

F. Proof of Proposition 2

Proof. Let $P(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{u}\|^2 + \frac{\lambda}{2}(\sum_{k=1}^t \|\boldsymbol{\omega}_k\|)^2 + \frac{1}{n}\sum_{i=1}^n \xi_i$, $Q(\mathbf{w}) = P(\mathbf{w}) + \frac{\sigma}{2}\|\boldsymbol{\omega}\|^2$ and $\Theta = \frac{1}{n}\sum_{i=1}^n(\max_{\mathbf{y}\neq\mathbf{y}_i} \Delta(\mathbf{y}, \mathbf{y}_i)) = P(\mathbf{0})$. Suppose $\bar{\mathbf{w}}$ is a minimizer of $P(\mathbf{w})$. Then, we have $P(\bar{\mathbf{w}}) \leq P(\mathbf{0})$. Accordingly, we have $\frac{\lambda}{2}||\bar{\boldsymbol{\omega}}||^2 \leq \frac{\lambda}{2}(\sum_{k=1}^t \|\bar{\boldsymbol{\omega}}_k\|)^2 \leq \frac{\lambda}{2}(\sum_{k=1}^t \|\bar{\boldsymbol{\omega}}_k\|)^2 + \frac{\lambda}{2}\|\bar{\mathbf{u}}\|^2 \leq P(\bar{\mathbf{w}}) \leq \Theta$, which implies that $\frac{\lambda}{2}||\bar{\boldsymbol{\omega}}||^2 \leq \Theta$. Let $\mathbf{w}^*$ be an $\frac{\epsilon}{2}$-accurate solution of (11). Then, we have $Q(\mathbf{w}^*) \leq Q(\bar{\mathbf{w}}) + \frac{\epsilon}{2}$. It follows that

$$P(\mathbf{w}^*) \leq Q(\mathbf{w}^*) \leq Q(\bar{\mathbf{w}}) + \frac{\epsilon}{2} = P(\bar{\mathbf{w}}) + \frac{\sigma}{2}\|\bar{\boldsymbol{\omega}}\|^2 + \frac{\epsilon}{2}.$$

By setting $\sigma \leq \lambda\epsilon/2\Theta$, we have $\frac{\sigma}{2}\|\bar{\boldsymbol{\omega}}\|^2 \leq \frac{\epsilon}{2}$, and $\mathbf{w}^*$ is an ϵ-accurate solution of (10). □

G. Proof of Theorem 2

The proof can be adapted from the proof of Corollary 3 in [29].