Journal of Machine Learning Research 14 (2013) 729-758 Submitted 4/11; Revised 9/12; Published 3/13

Semi-Supervised Learning Using Greedy Max-Cut

Jun Wang WANGJUN@US.IBM.COM
IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598, USA

Tony Jebara JEBARA@CS.COLUMBIA.EDU
Department of Computer Science
Columbia University
New York, NY 10027, USA

Shih-Fu Chang SFCHANG@EE.COLUMBIA.EDU
Department of Electrical Engineering
Columbia University
New York, NY 10027, USA

Editor: Mikhail Belkin

Abstract

Graph-based semi-supervised learning (*SSL*) methods play an increasingly important role in practical machine learning systems, particularly in agnostic settings when no parametric information or other prior knowledge is available about the data distribution. Given the constructed graph represented by a weight matrix, transductive inference is used to propagate known labels to predict the values of all unlabeled vertices. Designing a robust label diffusion algorithm for such graphs is a widely studied problem and various methods have recently been suggested. Many of these can be formalized as regularized function estimation through the minimization of a quadratic cost. However, most existing label diffusion methods minimize a univariate cost with the classification function as the only variable of interest. Since the observed labels seed the diffusion process, such univariate frameworks are extremely sensitive to the initial label choice and any label noise. To alleviate the dependency on the initial observed labels, this article proposes a bivariate formulation for graph-based *SSL*, where both the binary label information and a continuous classification function are arguments of the optimization. This bivariate formulation is shown to be equivalent to a linearly constrained Max-Cut problem. Finally an efficient solution via greedy gradient Max-Cut (*GGMC*) is derived which gradually assigns unlabeled vertices to each class with minimum connectivity. Once convergence guarantees are established, this greedy Max-Cut based *SSL* is applied on both artificial and standard benchmark data sets where it obtains superior classification accuracy compared to existing state-of-the-art *SSL* methods. Moreover, *GGMC* shows robustness with respect to the graph construction method and maintains high accuracy over extensive experiments with various edge linking and weighting schemes.

Keywords: graph transduction, semi-supervised learning, bivariate formulation, mixed integer programming, greedy Max-Cut

1. Introduction

In many real applications, labeled samples are scarce but unlabeled samples are abundant. Paradigms that consider both labeled and unlabeled data, that is, semi-supervised learning (*SSL*) methods, have

been increasingly explored in practical machine learning systems. While many semi-supervised learning approaches estimate a smooth function over labeled and unlabeled examples, this article presents a novel approach which emphasizes a *bivariate* optimization problem over the classification function *and* the labels. Prior to describing the method in detail, we briefly mention other *SSL* methods and previous work to motivate this article's contributions.

One of the earliest examples of the empirical advantages of *SSL* was co-training, a method first developed for text mining problems (Blum and Mitchell, 1998) and later extended in various forms to other applications (Chawla and Karakoulas, 2005; Goldman and Zhou, 2000). Therein, multiple classifiers are first estimated using conditionally independent feature sets of training data. The performance advantages of this method rely heavily on the existence of independent and complementary classifiers. Theoretical results show that some mild assumptions on the underlying data distribution are sufficient for co-training to work (Balcan et al., 2005; Wang and Zhou, 2010). However, performance can dramatically degrade if the classifiers do not complement each other or the independence assumption does not hold (Krogel and Scheffer, 2004). Though co-training is conceptually similar to semi-supervised learning due to the way it incorporates unlabeled data, the classifier training procedure itself is often supervised.

The extension of traditional supervised support vector machines (*SVMs*) to the semi-supervised scenario is another widely used *SSL* algorithm. Instead of maximizing separation (via a maximum-margin hyperplane) over training data as in standard *SVMs*, semi-supervised *SVMs* (*S3VMs*) estimate a hyperplane to balance maximum-margin partitioning of labeled data while encouraging a separation through low-density regions of the data (Vapnik, 1998). For example, transductive support vector machines (TSVMs) were developed as one of the earliest incarnations of semi-supervised *SVMs* (Joachims, 1999).[1] Various optimization techniques have been applied to solve *S3VMs* (Chapelle et al., 2008), resulting in a wide range of methods, such as low density separation (Chapelle and Zien, 2005), semi-definite programming based methods (Bie and Cristianini, 2004; Xu et al., 2008), and a branch-and-bound based approach (Chapelle et al., 2007).

Another family of *SSL* methods known as graph-based approaches have recently become popular due to their high accuracy and computational efficiency. Graph-based semi-supervised learning (*GSSL*) treats both labeled and unlabeled samples from a data set as vertices in a graph and builds pairwise edges between these vertices which are weighted by the affinity between the corresponding samples. The small portion of vertices with labels are then used by *SSL* methods to perform graph partition or information propagation to predict labels for unlabeled vertices. For instance, the graph mincuts approach formulates the label prediction as a graph cut problem (Blum and Chawla, 2001; Blum et al., 2004). Other *GSSL* methods, like graph transductive learning, formulate the problem as regularized function estimation over an undirected weighted graph. These methods optimize a trade-off between the accuracy of the classification function on labeled samples and a regularization term that favors a smooth function. The weighted graph and the optimal function ultimately propagate label information from labeled data to unlabeled data to produce transductive predictions. Popular algorithms for *GSSL* include graph cuts (Blum and Chawla, 2001; Blum et al., 2004; Joachims, 2003; Kveton et al., 2010), graph random walks (Azran, 2007; Szummer and Jaakkola, 2002), manifold regularization (Belkin et al., 2005, 2006; Sindhwani et al., 2008, 2005), and graph regularization (Zhou et al., 2004; Zhu et al., 2003). Comprehensive survey articles have also been disseminated (Zhu, 2005).

1. It is actually more appropriate to call this method a semi-supervised *SVM* since the learned classifier is indeed inductive (Zhu and Goldberg, 2009).

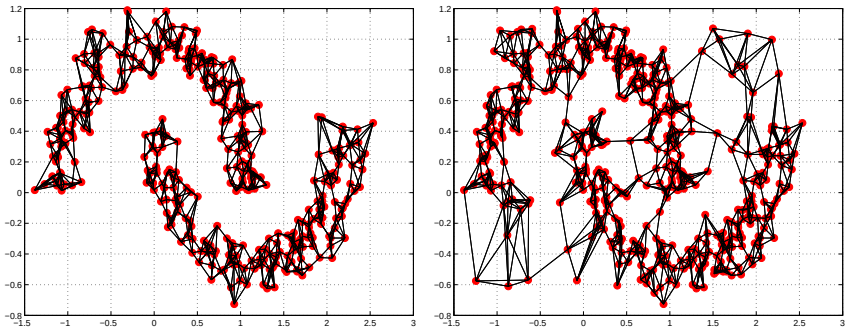
For some synthetic and real data problems, *GSSL* approaches do achieve promising performance. However, previous research has identified several realistic settings and labeling situations where this performance can be compromised (Wang et al., 2008b). In particular, both the graph construction methodology and the label initialization conditions can significantly impact prediction accuracy (Jebara et al., 2009). For a well-constructed graph such as the one shown in Figure 1(a), many *GSSL* methods produce satisfactory predictions. However, for graphs involving non-separable manifold structure as shown in Figure 1(b), prediction accuracy may deteriorate. Even if one assumes that the graph structures used in the above methods faithfully describe the data manifold, *GSSL* algorithms may still be misled by problems in the label information. Figure 3 depicts several cases where the label information leads to invalid graph transduction solutions for all the aforementioned algorithms.

In order to handle such challenging labeling conditions, we first extend the existing *GSSL* formulation by casting it as a *bivariate* optimization problem over the classification function *and* the labels. Then we demonstrate that minimizing the mixed bivariate cost function can be reduced to a pure integer programming problem that is equivalent to a constrained Max-Cut problem. Though semi-definite programming can be used to obtain approximate solutions, these are impractical due to scalability issues. Instead, an efficient greedy gradient Max-Cut (*GGMC*) solution is developed which remedies the instability previous methods seem to have vis-a-vis the initial labeling conditions on the graph. In the proposed greedy solution, initial labels simply act as initial values of the graph cut which is incrementally refined until convergence. During each iteration of the greedy search, the optimal unlabeled vertex is assigned to the labeled subset with minimum connectivity to maximally preserve cross-subset edge weight. Finally, an overall cut is produced after placing the unlabeled vertices into one of the label sets. It is then straightforward to obtain the final label prediction from the graph cut result. Note that this greedy gradient Max-Cut solution is equivalent to alternating between minimization of the cost over the label matrix and minimization of the cost over the prediction function. Moreover, to alleviate dependencies on the initialization of the cut (the given labels), a re-weighting of the connectivity between unlabeled vertices and labeled subsets is proposed. This re-weighting performs a within-class normalization using vertex degree as well as a between-class normalization using class prior information. We demonstrate that the greedy gradient Max-Cut based graph transduction produces significantly better performance on both artificial and real data sets.

The remainder of this paper is organized as the follows. Section 2 provides a brief background of graph-based *SSL* and discusses some open issues. In Section 3, we present our bivariate graph transduction framework, followed by the theoretical proof of its equivalence with the constrained Max-Cut problem in Section 4. In addition, a greedy gradient Max-Cut algorithm is proposed. Section 5 provides experimental validation for the algorithm on both toy and real classification data sets. Comparisons with leading semi-supervised methods are made. Concluding remarks and discussions are then provided in Section 6.

2. Background and Open Issues

In this section, we introduce some notation and then revisit two critical components of graph-based *SSL*: graph construction and label propagation. Subsequently, we discuss some challenging issues such as *SSL*'s sensitivity to graph construction and label initialization.



(a) (b)

Figure 1: Examples of constructed k-nearest-neighbors (kNN) graphs with $k = 5$ on the artificial two moon data set for a) the completely separable case; and b) the non-separable case with noisy samples.

2.1 Notations

We first summarize the notation used in this article. Assume we are given *iid* (independent and identically distributed) labeled samples $\{(\mathbf{x}_1, z_1), \ldots, (\mathbf{x}_l, z_l)\}$ as well as unlabeled samples $\{\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}\}$ drawn from a distribution $p(\mathbf{x}, z)$. Define the set of labeled inputs as $\mathbf{X}_l = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ with cardinality $|\mathbf{X}_l| = l$ and the set of unlabeled inputs $\mathbf{X}_u = \{\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}\}$ with cardinality $|\mathbf{X}_u| = u$. The labeled set $\mathbf{X}_l$ is associated with labels $\mathcal{Z}_l = \{z_1, \cdots, z_l\}$, where $z_i \in \{1, \cdots, c\}, i = 1, 2, \cdots, l$. The goal of semi-supervised learning is to infer the missing labels $\{z_{l+1}, \cdots, z_n\}$ corresponding to the unlabeled data $\{\mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\}$, where typically $l << n$ ($l + u = n$). A crucial component of *GSSL* is the estimation of a weighted sparse graph $\mathcal{G}$ from the input data $\mathbf{X} = \mathbf{X}_l \cup \mathbf{X}_u$. Subsequently, a labeling algorithm uses $\mathcal{G}$ and the known labels $\mathcal{Z}_l = \{z_1, \ldots, z_l\}$ to provide estimates $\hat{\mathcal{Z}}_u = \{\hat{z}_{l+1}, \ldots, \hat{z}_{l+u}\}$ which try to approximate the true labels $\mathcal{Z}_u = \{z_{l+1}, \ldots, z_{l+u}\}$ as measured by an appropriately chosen loss function.

In this article, assume the undirected graph converted from the data $\mathbf{X}$ is represented by $\mathcal{G} = \{\mathbf{X}, \mathbf{E}\}$, where the set of vertices is $\mathbf{X} = \{\mathbf{x}_i\}$ and the set of edges is $\mathbf{E} = \{e_{ij}\}$. Each sample $\mathbf{x}_i$ is treated as a vertex and the weight of edge e_{ij} is w_{ij}. Typically, one uses a kernel function $k(\cdot)$ over pairs of points to compute weights. The weights for edges are used to build a weight matrix which is denoted by $\mathbf{W} = \{w_{ij}\}$. Similarly, the vertex degree matrix $\mathbf{D} = diag([d_1, \cdots, d_n])$ is defined as $d_i = \sum_{j=1}^{n} w_{ij}$. The graph Laplacian is defined as $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$ and the normalized graph Laplacian is

$$\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{\Delta}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}.$$

The graph Laplacian and its normalized version can be viewed as operators on the space of functions f which can be used to define a regularization measure of smoothness over strongly-connected regions in a graph (Chung and Biggs, 1997). For example, the smoothness measurement of functions f using $\mathbf{L}$ over a graph is defined as

$$\langle f, \mathbf{L}f \rangle = \sum_i \sum_j w_{ij} \left\| \frac{f(\mathbf{x}_i)}{\sqrt{d_i}} - \frac{f(\mathbf{x}_j)}{\sqrt{d_j}} \right\|^2.$$

Finally, the label information is formulated as a label matrix $\mathbf{Y} = \{y_{ij}\} \in \mathbb{B}^{n\times c}$, where $y_{ij} = 1$ if sample $\mathbf{x}_i$ is associated with label j for $j \in \{1,2,\cdots,c\}$, that is, $z_i = j$, and $y_{ij} = 0$ otherwise. For single label problems (as opposed to multi-label problems), the constraints $\sum_{j=1}^{c} y_{ij} = 1$ are also imposed. Moreover, we will often refer to row and column vectors of such matrices, for instance, the i'th row and j'th column vectors of $\mathbf{Y}$ are denoted as $\mathbf{Y}_{i\cdot}$ and $\mathbf{Y}_{\cdot j}$, respectively. Let $\mathbf{F} = f(\mathbf{X})$ be the values of classification function over the data set $\mathbf{X}$. Most of the *GSSL* methods then use the graph quantity $\mathbf{W}$ as well as the known labels to recover a continuous classification function $\mathbf{F} \in \mathbb{R}^{n\times c}$ by minimizing a predefined cost on the graph.

2.2 Graph Construction for Semi-Supervised Learning

To estimate $\hat{\mathcal{Z}}_u = \{\hat{z}_{l+1},\ldots,\hat{z}_{l+u}\}$ using $\mathcal{G}$ and the known labels $\mathcal{Z}_l = \{z_1,\ldots,z_l\}$, we first convert the data points $\mathbf{X} = \mathbf{X}_l \cup \mathbf{X}_u$ into a graph $\mathcal{G} = \{\mathbf{X},\mathbf{E},\mathbf{W}\}$. This section discusses the graph construction method, $\mathbf{X} \to \mathcal{G}$, in detail. Given input data $\mathbf{X}$ with cardinality $|\mathbf{X}| = l+u$, graph construction produces a graph $\mathcal{G}$ consisting of $n = l+u$ vertices where each vertex is associated with the sample $\mathbf{x}_i$. The estimation of $\mathcal{G}$ from $\mathbf{X}$ usually proceeds in two steps.

The first step is to compute a score between all pairs of vertices using a similarity function. This creates a full adjacency matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$, where $\mathbf{K}_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$ is computed using kernel function $k(\cdot)$ to measure sample similarity. Subsequently, in the second step of graph construction, the matrix $\mathbf{K}$ is sparsified and reweighted to produce the final matrix $\mathbf{W}$. Sparsification is important since it leads to improved efficiency, better accuracy, and robustness to noise in the label inference stage. Furthermore, the kernel function $k(\cdot)$ is often only locally useful as a similarity and does not recover reliable weights between pairs of samples that are relatively far apart.

2.2.1 GRAPH SPARSIFICATION

Starting with the fully connected matrix $\mathbf{K}$, sparsification removes edges by recovering a binary matrix $\mathbf{B} \in \mathbb{B}^{n\times n}$ where $\mathbf{B}_{ij} = 1$ indicates that an edge is present between sample $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\mathbf{B}_{ij} = 0$ indicates the edge is absent (assume $\mathbf{B}_{ii} = 0$ unless otherwise noted). Here we will primarily investigate two graph sparsification algorithms: neighborhood approaches including the k-nearest and ε neighbors algorithms, and matching approaches such as b-matching (BM) (Edmonds and Johnson, 2003). All such methods operate on the matrix $\mathbf{K}$ or, equivalently, the distance matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$ obtained from $\mathbf{K}$ element-wise as $\mathbf{H}_{ij} = \sqrt{\mathbf{K}_{ii} + \mathbf{K}_{jj} - 2\mathbf{K}_{ij}}$.

Sparsification via Neighborhood Methods: There are two typical ways to build a neighborhood graph: the ε-neighborhood graph connecting samples within a distance of ε, and the kNN (k-nearest-neighbors) graph connecting k closest samples. Recent studies show the dramatic influences that different neighborhood methods have on clustering techniques (Carreira-Perpinán and Zemel, 2005; Maier et al., 2009). In practice, the kNN graph remains a more common approach since it is more adaptive to scale variation and data density anomalies while an improper threshold value in the ε-neighborhood graph may result in disconnected components or subgraphs in the data set or even isolated singleton vertices, as shown in Figure 2(b). In this article, we often use kNN neighborhood graphs since the ε-neighborhood graphs provide consistently weaker performance. In the remainder of this article, we will use neighborhood and kNN neighborhood graph interchangeably without specific declaration.

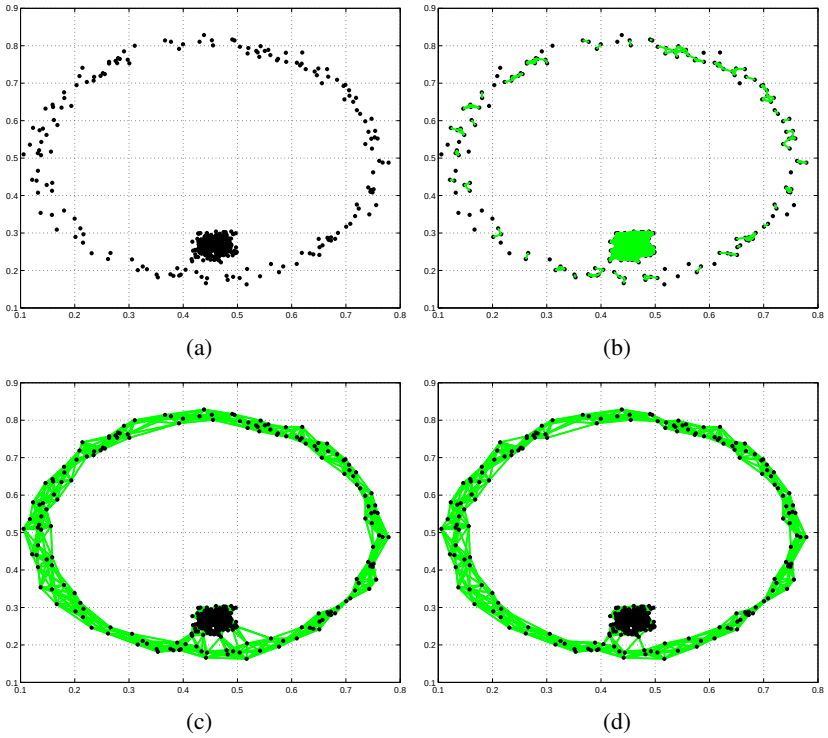


Figure 2: The synthetic data set used for demonstrating different graph construction approaches. a) The synthetic data; b) The ε-nearest neighbor graph; c) The k-nearest neighbor graph; d) The b-matched graph.

More specifically, the k-nearest neighbor graph is a graph in which two vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected by an edge if the distance $\mathbf{H}_{ij}$ between $\mathbf{x}_i$ and $\mathbf{x}_j$ is within or equal k-th smallest among the distances from $\mathbf{x}_i$ to other samples in $\mathbf{X}$. Roughly speaking, the k-nearest neighbors algorithm starts with a matrix $\hat{\mathbf{B}}$ of all zeros and for each point, searches for the k closest points to it (without considering itself). If a point j is one of the k closest neighbors to i, then we set $\hat{\mathbf{B}}_{ij} = 1$. It is straightforward to show that k-nearest neighbors search solves the following optimization problem:

$$\begin{aligned} &\min_{\hat{\mathbf{B}} \in \mathbb{B}} \sum_{ij} \hat{\mathbf{B}}_{ij} \mathbf{H}_{ij} \\ &s.t. \sum_j \hat{\mathbf{B}}_{ij} = k, \hat{\mathbf{B}}_{ii} = 0, \forall i, j \in 1, \ldots, n. \end{aligned} \tag{1}$$

The final solution of Equation (1) is produced by symmetrizing $\hat{\mathbf{B}}$ as follows $\mathbf{B}_{ij} = \max(\hat{\mathbf{B}}_{ij}, \hat{\mathbf{B}}_{ji})$.[2] This greedy algorithm is in fact not solving a well defined optimization problem over symmetric binary matrices. In addition, since it produces a symmetric matrix only via the *ad hoc* maximization over $\hat{\mathbf{B}}$ and its transpose, the solution $\mathbf{B}$ it produces does not satisfy the equality $\sum_k \mathbf{B}_{ij} = k$, but, rather, only satisfies the inequality $\sum_j \mathbf{B}_{ij} \geq k$. Ironically, despite conventional wisdom and the nomenclature, the k-nearest neighbors algorithm is producing an undirected subgraph with more

2. It is possible to replace the maximization operator with minimization to produce a symmetric matrix, yet in the setting $\mathbf{B} = \min(\hat{\mathbf{B}}, \hat{\mathbf{B}}^\top)$ the solution $\mathbf{B}$ only satisfies the inequality $\sum_j \mathbf{B}_{ij} \leq k$ and not the desired equality.

than k neighbors for each vertex. This motivates researchers to investigate the b-matching algorithm which actually achieves the desired output.

Sparsification via b-Matching: The b-matching problem generalizes maximum weight matching, that is, the linear assignment problem, where the objective is to find the binary matrix to minimize the optimization problem

$$\begin{aligned} &\min_{\mathbf{B}\in\mathbb{B}} \sum_{ij} \mathbf{B}_{ij}\mathbf{H}_{ij} \\ &s.t. \sum_j \mathbf{B}_{ij} = b, \mathbf{B}_{ii} = 0, \mathbf{B}_{ij} = \mathbf{B}_{ji}, \forall i, j \in 1,\ldots,n. \end{aligned} \tag{2}$$

achieving symmetry directly without post-processing. Here, the symmetric solution is recovered up-front by enforcing the additional constraints $\mathbf{B}_{ij} = \mathbf{B}_{ji}$. The matrix then satisfies the equality $\sum_j \mathbf{B}_{ij} = \sum_i \mathbf{B}_{ij} = b$ strictly. The solution to Equation (2) is not quite as straightforward or efficient as the greedy k-nearest neighbors algorithm. A polynomial time $O(bn^3)$ solution has been known, yet recent advances show that much faster alternatives are possible via (guaranteed) loopy belief propagation (Huang and Jebara, 2007).

Compared with the neighborhood graphs, the b-matching graph is balanced or b-regular. In other words, each vertex in the b-matched graph has exactly b edges connecting it to other vertices. This advantage plays a key role when conducting label propagation on typical samples $\mathbf{X}$ which are unevenly and non-uniformly distributed. Our previous work applied b-matching to construct graphs for semi-supervised learning tasks and demonstrated the superior performance over some unevenly sampled data (Jebara et al., 2009). For example, in Figure 2, this data set clearly contains two clusters of points, a dense Gaussian cluster surrounded by a ring cluster. Furthermore, the cluster data is unevenly sampled; one cluster is dense and the other is fairly sparse. In this example, the k-nearest neighbor graph constantly generates many cross-cluster edges while b-matching efficiently alleviates this problem by removing most of the improper edges. The example clearly shows that the b-matching technique produces regular graphs which could overcome the drawback of cross-structure linkages often generated by nearest neighbor methods. This intuitive study confirms the importance of graph construction methods and advocates b-matching as a valuable alternative to k-nearest neighbors, a method that many practitioners expect to produce regular undirected graphs, though in practice often generates irregular graphs.

2.2.2 GRAPH EDGE RE-WEIGHTING

Once a graph has been sparsified and a binary matrix $\mathbf{B}$ is computed and used to delete unwanted edges, several procedures can then be used to update the weights in the matrix $\mathbf{K}$ to produce a final set of edge weights $\mathbf{W}$. Specifically, whenever $\mathbf{B}_{ij} = 0$, the edge weight is also $w_{ij} = 0$; however, $\mathbf{B}_{ij} = 1$ implies that $w_{ij} \geq 0$. Two popular approaches are considered here for estimating the non-zero components of $\mathbf{W}$.

Binary Weighting: The simplest approach for building the weighted graph is the *binary* weighting approach, where all the linked edges in the graph are given the weight 1 and the edge weights of disconnected vertices are given the weight 0. In other words, this setting simply uses $\mathbf{W} = \mathbf{B}$. However, this uniform weight on graph edges can be sensitive, particularly if some of the graph vertices were improperly connected by the sparsification procedure (either the neighborhood based procedures or the b-matching procedure).

Gaussian Kernel Weighting: An alternative approach is *Gaussian kernel* weighting which is often applied to modulate sample similarity. Therein, the edge weight between two connected

samples $\mathbf{x}_i$ and $\mathbf{x}_j$ is computed as:

$$w_{ij} = \mathbf{B}_{ij} \exp\left(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma^2}\right),$$

where the function $d(\mathbf{x}_i, \mathbf{x}_j)$ evaluates the dissimilarity of samples $\mathbf{x}_i$ and $\mathbf{x}_j$, and σ is the kernel bandwidth parameter. There are many choices for the distance function $d(\cdot)$ including any ℓ_p distance, χ^2 distance, and cosine distance (Zhu, 2005; Belkin et al., 2005; Jebara et al., 2009).

This final step in the graph construction procedure ensures that the unlabeled data $\mathbf{X}$ has now been converted into a graph $\mathcal{G}$ with a weighted sparse undirected adjacency matrix $\mathbf{W}$. Given this graph and some initial label information $\mathbf{Y}_l$, any of the current popular algorithms for graph based SSL can be used to solve the labeling problem.

2.3 Univariate Graph Regularization Framework

Given the constructed graph $\mathcal{G} = \{\mathbf{X}, \mathbf{E}\}$, whose geometric structure is represented by the weight matrix $\mathbf{W}$, the label inference task is to diffuse the known labels $\mathcal{Z}_l$ to all the unlabeled vertices $\mathbf{X}_u$ in the graph and estimate $\hat{\mathcal{Z}}_u$. Designing a robust label diffusion algorithm for such graphs is a widely studied problem (Chapelle et al., 2006; Zhu, 2005; Zhu and Goldberg, 2009).

Here we are particularly interested in a category of approaches, which estimate the prediction function $\mathbf{F} \in \mathbb{R}^{n \times c}$ by minimizing a quadratic cost defined over the graph. The cost function typically involves a trade-off between the smoothness of the function over the graph of both labeled and unlabeled data (consistency of the predictions on closely connected vertices) and the accuracy of the function at fitting the label information on the labeled vertices. Approaches like the Gaussian fields and harmonic functions (*GFHF*) method (Zhu et al., 2003) and the local and global consistency (*LGC*) method (Zhou et al., 2004) fall into this category, so does our previous method of graph transduction via alternating minimization (Wang et al., 2008b).

Both *LGC* and *GFHF* define a cost function Q that involves the combined contribution of two penalty terms: the global smoothness Q_{smooth} and local fitting accuracy Q_{fit}. The final prediction function $\mathbf{F}$ is obtained by minimizing the cost function as:

$$\mathbf{F}^* = \arg\min_{\mathbf{F} \in \mathbb{R}^{n \times c}} Q(\mathbf{F}) = \arg\min_{\mathbf{F} \in \mathbb{R}^{n \times c}} \left(Q_{smooth}(\mathbf{F}) + Q_{fit}(\mathbf{F})\right). \tag{3}$$

A natural formulation of the above cost function is *LGC* (Zhou et al., 2004) which uses an elastic regularizer framework as follows

$$Q(\mathbf{F}) = \|\mathbf{F}\|_{\mathcal{G}}^2 + \frac{\mu}{2}\|\mathbf{F} - \mathbf{Y}\|^2. \tag{4}$$

The first term $\|\mathbf{F}\|_{\mathcal{G}}^2$ represents function smoothness over graph $\mathcal{G}$ and $\|\mathbf{F} - \mathbf{Y}\|^2$ measures the empirical loss on given labeled samples. Specifically, in *LGC*, the function smoothness is defined using the semi-inner product

$$Q_{smooth} = \|\mathbf{F}\|_{\mathcal{G}}^2 = \frac{1}{2}\langle \mathbf{F}, \mathbf{L}\mathbf{F} \rangle = \frac{1}{2}\mathrm{tr}(\mathbf{F}^\top \mathbf{L}\mathbf{F}).$$

Note that the coefficient μ in Equation (4) balances global smoothness and local fitting terms. If we set $\mu = \infty$ and use a standard graph Laplacian quantity $\mathbf{\Delta}$ for the smoothness term, the above

framework reduces to the harmonic function formulation (Zhu et al., 2003). More precisely, the cost function only preserves the smoothness term as

$$Q(\mathbf{F}) = \mathrm{tr}(\mathbf{F}^\top \boldsymbol{\Delta} \mathbf{F}). \tag{5}$$

Meanwhile, the harmonic function $\mathbf{F}$ minimizing the above cost also satisfies two conditions:

$$\begin{aligned} &\frac{\partial Q}{\partial \mathbf{F}}_u = \boldsymbol{\Delta}\mathbf{F}_u = 0, \\ &\mathbf{F}_l = \mathbf{Y}_l, \end{aligned}$$

where $\mathbf{F}_l, \mathbf{F}_u$ are the function values of $f(\cdot)$ over labeled and unlabeled vertices, that is, $\mathbf{F}_l = f(\mathbf{X}_l)$, $\mathbf{F}_u = f(\mathbf{X}_u)$, and $\mathbf{F} = [\mathbf{F}_l \ \ \mathbf{F}_u]^\top$. The first equation above denotes the zero derivative of the object function on the unlabeled data and the second equation clamps the function value on the given label value $\mathbf{Y}_l$. Both *LGC* and *GFHF* are univariate regularization frameworks where the continuous prediction function is treated as the only variable in the optimization procedure. The optimal solutions for Equation (4) and Equation (5) are easily obtained by solving a linear system.

2.4 Open Issues

Existing graph-based *SSL* methods hinge on having good label information and an appropriately constructed graph (Wang et al., 2008b; Liu et al., 2012). But the heuristic design of the graph may result in suboptimal inference. In addition, the label propagation procedure can easily be misled if there exist excessive noise or outliers in the initial labeled set. Finally, in *iid* settings, the difference between empirically estimated class proportions and their true expected value is bounded (Huang and Jebara, 2010). However, practical annotation procedures are not necessarily *iid* and labeled data may have empirical class frequencies that deviate significantly from the expected class ratios. These degenerate situations seem to plague real world problems and compromise the performance of many state-of-the-art *SSL* algorithms. We next discuss some open issues which occur often in graph construction and label propagation, two critical components of all *GSSL* algorithms.

2.4.1 SENSITIVITY TO GRAPH CONSTRUCTION

As shown in Figure 1(a), a well-built graph obtained from separable manifolds of data will achieve good results with most existing *GSSL* approaches. However, practical applications often produce non-separable graphs as shown in Figure 1(b). In addition to the widely used kNN graph, we showed that b-matching could be used successfully for graph construction (Jebara et al., 2009). But both kNN graphs and b-matched graphs are heuristics and require the careful selection of the parameter k or b which controls the number of links incident to each vertex in the graph. Moreover, edge reweighing on the sparse graph often also requires exploration forcing the user to select kernels and various kernel parameters. All these heuristic steps in graph design require extra effort from the user and demand some level of familiarity with the data domain.

2.4.2 SENSITIVITY TO LABEL NOISE

Most of the existing *GSSL* methods are based on an univariate quadratic regularization framework which relies heavily on the quality of the initially assigned labels. For certain synthetic and real data problems, such graph transduction approaches achieve promising performance. However, several realistic labeling conditions produce unsatisfactory performance (Wang et al., 2008b). Even if

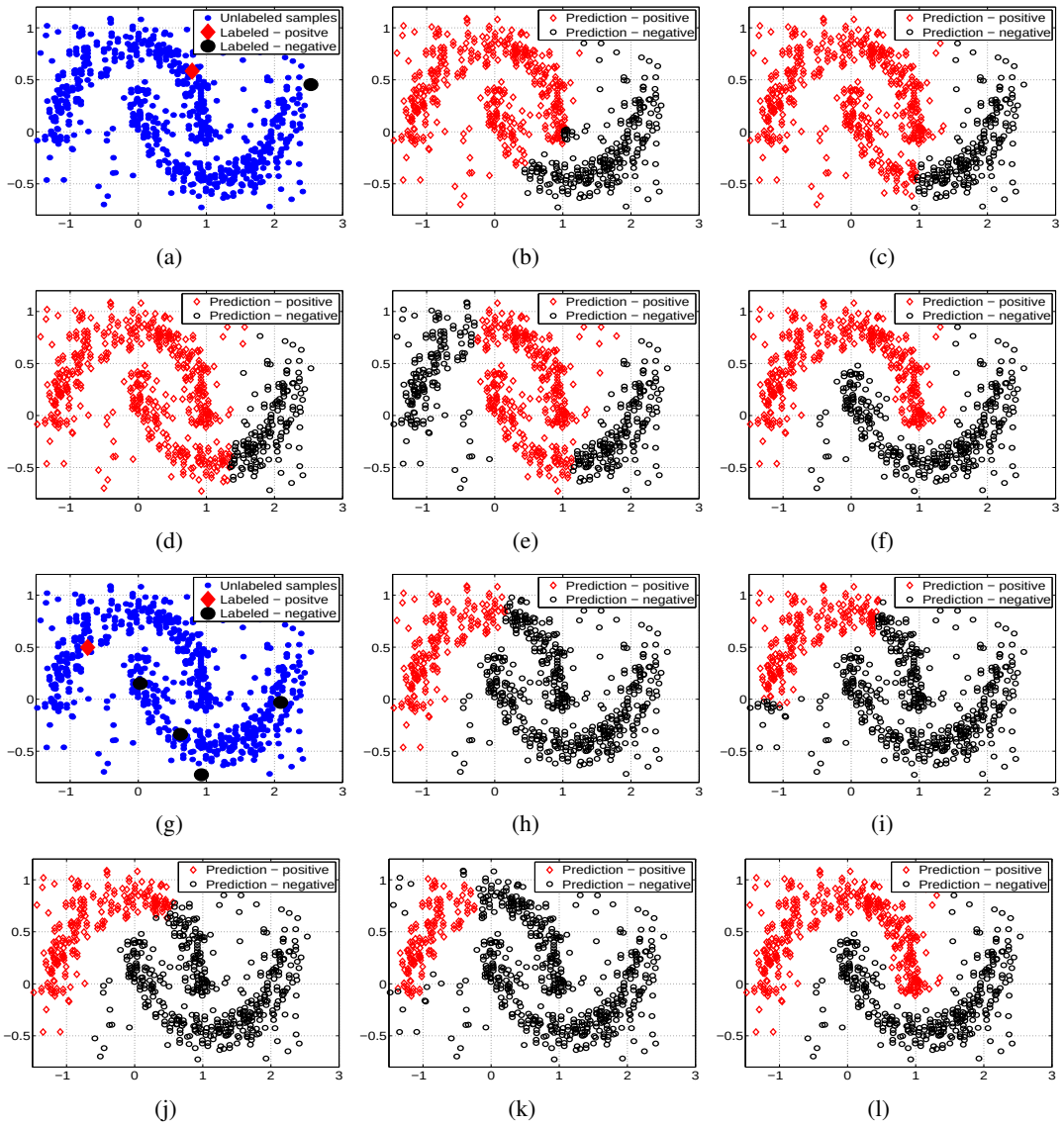


Figure 3: Examples illustrating the sensitivity of graph-based *SSL* to adverse labeling conditions. Particularly challenging conditions are shown in (a) where an uninformative label on an outlier sample is the only negative label (denoted by a black circle) and in (g) where imbalanced labeling is involved. Prediction results are shown for the *GFHF* method (Zhu et al., 2003) in (b) and (h), the *LGC* method (Zhou et al., 2004) in (c) and (i), the *LapSVM* method (Belkin et al., 2006) in (d) and (j), the *TSVM* method (Joachims, 1999) in (e) and (k); and our method in (f) and (l).

the graph is perfectly constructed from the data, problematic initial labels under practical situations can easily deteriorate the performance of *SSL* prediction. Figure 3 provides examples depicting imbalanced and noisy labels that lead to invalid graph transduction solutions for all the aforementioned algorithms. The first labeling problem involves uninformative labels (Figure 3(a)). The only

negative label (dark circle) is located in an outlier region where the low density connectivity limits its diffusion to the rest of the graph. The leading *SSL* methods classify the majority of unlabeled nodes in the graph as positive (Figure 3(b)-Figure 3(e)). Such conditions are frequent in real problems like content-based image retrieval (CBIR) where the visual query example is not necessarily representative of the class. Another difficult case is due to imbalanced labeling. There, the ratio of training labels is disproportionate to the underlying class proportions. For example, Figure 3(g) depicts two half-circles with an almost equal number of samples. However, since the training labels contain three negative samples and only one positive example, the *SSL* predictions are strongly biased towards the negative class (see Figures 3(h) to 3(k)). This imbalanced labeling situation occurs frequently in realistic problems such as the annotation of microscopic images (Wang et al., 2008a). Therein, the human labeler favors certain cellular phenotypes due to domain-specific biological hypotheses. To tackle these issues, we next propose a novel bivariate framework for graph-based *SSL* and describe an efficient algorithm that achieves it via alternating minimization.

3. Bivariate Framework for Graph-Based *SSL*

We first propose an extension to the existing graph regularization-based *SSL* formulations by casting the problem as a *bivariate* optimization over both the classification function and the unknown labels. Then we demonstrate that the minimization of this bivariate cost reduces to a linearly constrained binary integer programming (BIP) problem. This problem can be approximated via semi-definite programming yet this approach is impractical due to scalability issues. We instead explore a fast method which alternates minimization of the cost over the label matrix and the prediction function.

3.1 The Cost Function

Recall the univariate regularization formulation for graph-based *SSL* in Equation (3). Also note that the optimization problem in existing approaches such as *LGC* and *GFHF* can be broken up into separate parallel problems since the cost function decomposes into additive terms that only depend on individual columns of the prediction matrix $\mathbf{F}$ (Wang et al., 2008a). Such a decomposition reveals that biases may arise if the input labels are disproportionately imbalanced. In addition, when the graph contains background noise and makes class manifolds non-separable (as in Figure 1(b)), these existing graph transduction approaches fail to output reasonable classification results.

Since the univariate framework treats the initial label information as a constant, we propose a novel bivariate optimization framework that explicitly optimizes over both the classification function $\mathbf{F}$ and the binary label matrix $\mathbf{Y}$:

$$\begin{aligned}(\mathbf{F}^*,\mathbf{Y}^*) \quad &= \arg\min_{\mathbf{F}\in\mathbb{R}^{n\times c},\mathbf{Y}\in\mathbb{B}^{n\times c}} Q(\mathbf{F},\mathbf{Y}) \\ &s.t. \quad y_{ij}\in\{0,1\}, \\ &\quad \textstyle\sum_{j=1}^{c} y_{ij}=1, \\ &\quad y_{ij}=1, \ \text{for} \ \ z_i=j, \ \ j=1,\cdots,c.,\end{aligned}$$

where $\mathbb{B}^{n\times c}$ is the set of all binary matrices $\mathbf{Y}$ of size $n\times c$. For a labeled sample $\mathbf{x}_i\in\mathbf{X}_l$, $y_{ij}=1$ if $z_i=j$, and the constraint $\sum_{j=1}^{c} y_{ij}=1$ indicates that this a single label prediction problem. We specify the cost function as

$$Q(\mathbf{F},\mathbf{Y})=\frac{1}{2}\mathrm{tr}\left(\mathbf{F}^\top\mathbf{L}\mathbf{F}+\mu(\mathbf{F}-\mathbf{Y})^\top(\mathbf{F}-\mathbf{Y})\right). \tag{6}$$

Finally, rewriting the cost as a summation (Zhou et al., 2004) reveals a more intuitive formulation where

$$Q(\mathbf{F},\mathbf{Y}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\left\|\frac{\mathbf{F}_{i\cdot}}{\sqrt{d_i}} - \frac{\mathbf{F}_{j\cdot}}{\sqrt{d_j}}\right\|^2 + \frac{\mu}{2}\sum_{i=1}^{n}\|\mathbf{F}_{i\cdot}-\mathbf{Y}_{i\cdot}\|^2.$$

3.2 Reduction to a Univariate Problem

In the new graph regularization framework proposed above, the cost function involves two variables to be optimized. Simultaneously recovering both solutions is intractable due to the mixed integer programming problem over binary $\mathbf{Y}$ and continuous $\mathbf{F}$. To solve the issue, we first show how to reduce the original mixed problem to a univariate optimization problem with respect to the label variable $\mathbf{Y}$.

$\mathbf{F}$ *optimization step:*
In each loop with $\mathbf{Y}$ fixed, the classification function $\mathbf{F} \in \mathbb{R}^{n\times c}$ is continuous and the cost function is convex, allowing the minimum to be recovered by setting the partial derivative to zero:

$$\begin{aligned}&\frac{\partial Q}{\partial \mathbf{F}^*} = 0 \Longrightarrow \mathbf{L}\mathbf{F}^* + \mu(\mathbf{F}^*-\mathbf{Y}) = 0\\ &\Longrightarrow \mathbf{F}^* = (\mathbf{L}/\mu+\mathbf{I})^{-1}\mathbf{Y} = \mathbf{P}\mathbf{Y},\end{aligned} \tag{7}$$

where we denote the $\mathbf{P}$ matrix as

$$\mathbf{P} = (\mathbf{L}/\mu+\mathbf{I})^{-1},$$

and name it the *propagation matrix* since it is used to derived a prediction function $\mathbf{F}$ given a label matrix $\mathbf{Y}$. Because the graph is often symmetric, it is easy to show that the graph Laplacian $\mathbf{L}$ and the propagation matrix $\mathbf{P}$ are both symmetric.

$\mathbf{Y}$ *optimization step:*
Next replace $\mathbf{F}$ in Equation (6) by its optimal value $\mathbf{F}^*$ from the solution of Equation (7). This yields

$$\begin{aligned}Q(\mathbf{Y}) &= \frac{1}{2}\mathrm{tr}(\mathbf{Y}^\top\mathbf{P}^\top\mathbf{L}\mathbf{P}\mathbf{Y} + \mu(\mathbf{P}Y-\mathbf{Y})^\top(\mathbf{P}\mathbf{Y}-\mathbf{Y}))\\ &= \frac{1}{2}\mathrm{tr}\left(\mathbf{Y}^\top\left[\mathbf{P}^\top\mathbf{L}\mathbf{P} + \mu(\mathbf{P}^\top-\mathbf{I})(\mathbf{P}-\mathbf{I})\right]\mathbf{Y}\right) = \frac{1}{2}\mathrm{tr}\left(\mathbf{Y}^\top\mathbf{A}\mathbf{Y}\right),\end{aligned}$$

where we group all the constant parts in the above equation and define

$$\mathbf{A} = \mathbf{P}^\top\mathbf{L}\mathbf{P} + \mu(\mathbf{P}^\top-\mathbf{I})(\mathbf{P}-\mathbf{I}) = \mathbf{P}^\top\mathbf{L}\mathbf{P} + \mu(\mathbf{P}-\mathbf{I})^2.$$

The final optimization problem becomes

$$\begin{aligned}&\mathbf{Y}^* = \arg\min\frac{1}{2}\mathrm{tr}\left(\mathbf{Y}^\top\mathbf{A}\mathbf{Y}\right)\\ &s.t.\quad y_{ij}\in\{0,1\},\\ &\qquad \textstyle\sum_j y_{ij} = 1,\ \ j = 1,\cdots,c\\ &\qquad y_{ij} = 1, \text{for}\ \ z_i = j,\ \ j = 1,\cdots,c.\end{aligned} \tag{8}$$

The first constraint produces a binary integer problem and the second one $\sum_j y_{ij} = 1$ produces a single assignment constraint, that is, each vertex can only be assigned one class label. The third group of constraints encodes the initial label information in the variable $\mathbf{Y}$. Since the binary matrix $\mathbf{Y} \in \mathbb{B}^{n \times c}$ is subject to linear constraints of the form $\sum_j y_{ij} = 1$ and initial labeling conditions, the optimization in Equation (8) requires solving a linearly constrained binary integer programming (BIP) problem which is NP hard (Cook, 1971; Karp, 1972).

3.3 Incorporating Label Normalization

A straightforward approach to solving the minimization problem in Equation (8) is to use the gradient to greedily update the label variable $\mathbf{Y}$. However, this may produce biased classification results in practice since, at each iteration, the class with more labels will be preferred and will propagate more quickly to the unlabeled examples. This arises in practice (as in Figure 3) and is due to the fact that $\mathbf{Y}$ starts off sparse and contains many unknown entries. To compensate for this bias during label propagation, we propose using a normalized label variable $\tilde{\mathbf{Y}} = \mathbf{\Lambda} Y$ for computing the cost function in Equation (6) as

$$\begin{aligned} Q &= \frac{1}{2}\mathrm{tr}\left(\mathbf{F}^\top \mathbf{L}\mathbf{F} + \mu(\mathbf{F}-\tilde{\mathbf{Y}})^\top(\mathbf{F}-\tilde{\mathbf{Y}})\right) \\ &= \frac{1}{2}\mathrm{tr}\left(\mathbf{F}^\top \mathbf{L}\mathbf{F} + \mu(\mathbf{F}-\mathbf{\Lambda}\mathbf{Y})^\top(\mathbf{F}-\mathbf{\Lambda}\mathbf{Y})\right). \end{aligned} \tag{9}$$

The diagonal matrix $\mathbf{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda}) = \mathrm{diag}([\lambda_1, \cdots, \lambda_n])$ is introduced to re-weight or re-balance the influence of labels from different classes as it modulates the label importance based on node degree. The value of λ_i $(i = 1, \cdots, n)$ is computed using the vertex degree d_i and label information

$$\lambda_i = \begin{cases} p_j \cdot \frac{d_i}{\sum_k y_{kj} d_k} & : \quad y_{ij} = 1 \\ 0 & : \quad \text{otherwise}, \end{cases} \tag{10}$$

where p_j is the prior of class j and is subject to the constraint $\sum_{j=1}^{c} p_j = 1$. The value of p_j can be either estimated from the labeled training set or simply set to be uniform $p_j = 1/c$ $(j = 1, \cdots, c)$ in agnostic situations (when no better prior is available or if the labeled data is plagued by biased sampling). Using the normalized label matrix $\tilde{\mathbf{Y}}$ in the bivariate formulation allows labeled nodes with high degrees to contribute more during the label propagation process. However, the total diffusion of each class is kept equal (for agnostic settings with no priors available) or proportional to the class prior (for the setting with prior information). Therefore, the influence of different classes is balanced even if the given class labels are imbalanced. If class proportion information is known, it can be integrated by scaling the diffusion with the appropriate prior. In other words, the label normalization attempts to enforce simple concentration inequalities which, in the *iid* case require the predicted label results to concentrate around the underlying class ratios (Huang and Jebara, 2010). This intuition is in line with prior work that uses class proportion information in transductive inference where class proportion is enforced as a hard constraint (Chapelle et al., 2007) or as a regularizer (Mann and McCallum, 2007).

3.4 Alternating Minimization Procedure

To solve the above refined problem, we proposed an alternating minimization algorithm (Wang et al., 2008b). Briefly, starting with Equation (9) and repeating the similar derivation as in Section 3.2, we

obtain the optimal solution $\mathbf{F}^*$ and the final cost function with respect to label variable $\mathbf{Y}$ as

$$\mathbf{F}^* = \mathbf{P}\tilde{\mathbf{Y}} = \mathbf{P}\mathbf{\Lambda}\mathbf{Y}, \tag{11}$$

$$Q = \frac{1}{2}\mathrm{tr}\left(\tilde{\mathbf{Y}}^\top \mathbf{A}\tilde{\mathbf{Y}}\right) = \frac{1}{2}\mathrm{tr}\left(\mathbf{Y}^\top \mathbf{\Lambda}\mathbf{A}\mathbf{\Lambda}\mathbf{Y}\right). \tag{12}$$

Instead of finding the global optimum $\mathbf{Y}^*$, we only take an incremental step in each iteration to modify a single entry in $\mathbf{Y}$. Namely in each iteration, we find the optimal position (i^*, j^*) in the matrix $\mathbf{Y}$ and change the binary value of $y_{i^* j*}$ from 0 to 1. To do this, we find the direction with the largest negative gradient guiding our choice of binary step on $\mathbf{Y}$. Specifically, we evaluate $\| \bigtriangledown Q_{\mathbf{Y}} \|$ and find the largest negative value to determine (i^*, j^*).

Note that the setting $y_{i^* j^*} = 1$ is equivalent to modifying the normalized label matrix $\tilde{\mathbf{Y}}$ by setting $\tilde{y}_{i^*, j^*} = \varepsilon_{i^*}, 0 < \varepsilon_{i^*} < 1$, and $\mathbf{Y}, \tilde{\mathbf{Y}}$ can be converted from each other componentwise. Thus, the greedy optimization of Q with respect to $\mathbf{Y}$ is equivalent to greedy minimization of Q with respect to $\tilde{\mathbf{Y}}$. More formally, we derive the gradient of the above loss function $\nabla_{\tilde{\mathbf{Y}}} Q = \frac{\partial Q}{\partial \tilde{\mathbf{Y}}}$ and recover it with respect to $\mathbf{Y}$ as:

$$\frac{\partial Q}{\partial \tilde{\mathbf{Y}}} = \mathbf{A}\tilde{\mathbf{Y}} = \mathbf{A}\mathbf{\Lambda}\mathbf{Y}. \tag{13}$$

As described earlier, we search the gradient matrix $\nabla_{\tilde{\mathbf{Y}}} Q$ to find the minimal element

$$(i^*, j^*) = \arg\min_{\mathbf{x}_i \in \mathcal{X}_u, 1 \le j \le c} \nabla_{\tilde{y}_{ij}} Q.$$

Because of the binary nature of $\mathbf{Y}$, we simply set $y_{i^* j^*} = 1$ instead of making a continuous update. Accordingly, the node weight matrix $\mathbf{\Lambda}^{t+1}$ can be recalculated with the updated $\mathbf{Y}^{t+1}$ in the iteration $t+1$. The update of $\mathbf{Y}$ is greedy and thus it could backtrack from predicted labels in previous iterations without convergence guarantees. We propose a straightforward way to guarantee convergence and avoid backtracking or unstable oscillation in the greedy propagation process: once an unlabeled point has been labeled, its labeling can no longer be changed. Thus, we remove the most recently labeled point (i^*, j^*) from future consideration and conduct search over the remaining unlabeled data only. In other words, to avoid retroactively changing predicted labels, the labeled vertex $\mathbf{x}_{i^*}$ is removed from $\mathbf{X}_u$ and added to $\mathbf{X}_l$.

Note that although the optimal $\mathbf{F}^*$ can be computed using Equation (11), this need not be done explicitly. Instead, the new value is implicitly used in Equation (14) only to update $\mathbf{Y}$. In the following, we summarize the update rules from step t to $t+1$ in the alternating minimization scheme.

. Compute gradient matrix:

$$(\nabla_{\tilde{\mathbf{Y}}} Q)^t = \mathbf{A}\tilde{\mathbf{Y}}^t = \mathbf{A}\mathbf{\Lambda}^t\mathbf{Y}^t, \mathbf{\Lambda}^t = \mathrm{diag}(\boldsymbol{\lambda}^t).$$

. Update one label:

$$\begin{aligned} (i^*, j^*) &= \arg\min_{\mathbf{x}_i \in \mathbf{X}_u, 1 \le j \le c} (\nabla_{\tilde{y}_{ij}} Q)^t, \\ y_{i^* j^*}^{t+1} &= 1. \end{aligned}$$

. Update label normalization matrix:

$$\lambda^{t+1} \quad = \quad \begin{cases} \frac{\mathbf{D}_{ii}}{\sum_k \mathbf{Y}_{kj}^{t+1}\mathbf{D}_{kk}} & : \quad y_{ij}^{t+1} = 1 \\ 0 & : \quad \text{otherwise.} \end{cases}$$

. Update the list of labeled and unlabeled data:

$$\mathbf{X}_l^{t+1} \longleftarrow \mathbf{X}_l^t + \mathbf{x}_{i^*} \;\; ; \;\; \mathbf{X}_u^{t+1} \longleftarrow \mathbf{X}_u^t - \mathbf{x}_{i^*}.$$

Starting with a few given labels, the method iteratively and greedily updates the label matrix $\mathbf{Y}$ to derive new labels in each iteration. The newly obtained labels are then use in the next iteration. Notice that the label normalization vector is re-computed for each iteration due to the change of label set. Although the original objective is formed in a bivariate manner in Equation 6, the above alternating optimization procedure drives the prediction of new labels without explicitly calculating $\mathbf{F}^*$ as is done in other graph transduction methods like *LGC* and *GFHF*. This unique feature makes the proposed algorithm very efficient since we only update the gradient matrix $\nabla_{\tilde{\mathbf{Y}}} Q$ for prediction new labels in each iteration.

Due to the greedy assignment step and the lack of back-tracking, the algorithm can repeat the alternating minimization (or the gradient computation) at most $n-l$ times. Each minimization step over $\mathbf{F}$ and $\mathbf{Y}$ requires $O(n^2)$ complexity and, thus, the total complexity of the above greedy algorithm is $O(n^3)$. However, the update of the graph gradient can be done efficiently by modifying only a single entry in $\mathbf{Y}$ per iteration. This further reduces the computational cost down to $O(n^2)$. Empirically, the value of the loss function Q decreases rapidly in the the first dozen iterations and steadily converges afterward (Wang et al., 2009). This phenomenon indicates that early stopping strategy could be applied to speed up the training and prediction (Melacci and Belkin, 2011). Once the first few iterations are completed, the new labels are added and the standard propagation step can be used to predict the optimal $\mathbf{F}^*$ as indicated in Equation (11) over the whole graph in one step. The details of the algorithm, namely *graph transduction via alternating minimization*, can be referred to Wang et al. (2008b). In the following section, we provide a greedy Max-Cut based solution, which essentially interprets the above alternating minimization procedure from a graph cut view.

4. Greedy Max-Cut for Semi-Supervised Learning

In this section, we introduce a connection between the proposed bivariate graph transduction framework and the well-known maximum cut problem. Then, a greedy gradient based Max-Cut solution will be developed and related to the above alternating minimization algorithm.

4.1 Equivalence to a Constrained Max-Cut Problem

Recall the optimization problem defined in Equation (8) which is exactly a linearly constrained binary integer programming (BIP) problem. In the case of a two-class problem, this optimization will be reduced to a weighted Max-Cut problem over the graph $\mathcal{G}_{\mathbf{A}} = \{\mathbf{X}, \mathbf{A}\}$ subject to linear constraints. The cost function in Equation (8) can be rewritten as

$$Q(\mathbf{Y}) = \frac{1}{2}\mathrm{tr}\left(\mathbf{Y}^\top \mathbf{A}\mathbf{Y}\right) = \frac{1}{2}\mathrm{tr}\left(\mathbf{A}\mathbf{Y}\mathbf{Y}^\top\right) = \frac{1}{2}\mathrm{tr}\left(\mathbf{A}\mathbf{R}\right),$$

where $\mathbf{A} = \{a_{ij}\}$ and $\mathbf{R} = \mathbf{Y}\mathbf{Y}^\top$. Considering the constraints $\sum_j y_{ij} = 1$ and $\mathbf{Y} \in \mathbb{B}^{n\times 2}$ for a two-class problem, we let

$$\mathbf{Y} = [\mathbf{y} \;\; \mathbf{e}-\mathbf{y}],$$

where $\mathbf{y} \in \mathbb{B}^n$ (i.e., $\mathbf{y} = \{y_i\}, y_i \in \{0,1\}, i = 1,\cdots,n$) and $\mathbf{e} = [1,1,\cdots,1]^\top$ are column vectors. Then rewrite $\mathbf{R}$ as

$$\begin{aligned} \mathbf{R} &= \mathbf{Y}\mathbf{Y}^\top = [\mathbf{y} \;\; \mathbf{e}-\mathbf{y}][\mathbf{y} \;\; \mathbf{e}-\mathbf{y}]^\top \\ &= \mathbf{e}\mathbf{e}^\top - \mathbf{y}(\mathbf{e}^\top - \mathbf{y}^\top) - (\mathbf{e}-\mathbf{y})\mathbf{y}^\top. \end{aligned} \tag{14}$$

Now rewrite the cost function in Equation (14) by replacing $\mathbf{R}$ with Equation (14)

$$Q(\mathbf{y}) = \frac{1}{2}\mathrm{tr}\left(\mathbf{A}\left[\mathbf{e}\mathbf{e}^\top - \mathbf{y}(\mathbf{e}^\top - \mathbf{y}^\top) - (\mathbf{e}-\mathbf{y})\mathbf{y}^\top\right]\right).$$

Since $\mathbf{e}\mathbf{e}^\top$ is the all-ones matrix, we obtain

$$\frac{1}{2}\mathrm{tr}\left(\mathbf{A}\mathbf{e}\mathbf{e}^\top\right) = \frac{1}{2}\sum_i\sum_j \mathbf{A}_{ij}.$$

It is easy to show that $\mathbf{A}$ is symmetric and

$$\mathbf{y}(\mathbf{e}^\top - \mathbf{y}^\top) = [(\mathbf{e}-\mathbf{y})\mathbf{y}^\top]^\top.$$

Next, simplify the cost function Q as

$$\begin{aligned} Q(\mathbf{y}) &= \frac{1}{2}\mathrm{tr}\left(\mathbf{A}\mathbf{e}\mathbf{e}^\top\right) - \mathrm{tr}\left[(\mathbf{e}^\top - \mathbf{y}^\top)\mathbf{A}\mathbf{y}\right] \\ &= \frac{1}{2}\mathrm{tr}\left(\mathbf{A}\mathbf{e}\mathbf{e}^\top\right) - \mathbf{y}^\top\mathbf{A}(\mathbf{e}-\mathbf{y}). \end{aligned}$$

Since the first part is a constant, the optimal value $\mathbf{y}^*$ of the above minimization problem is the argument of the maximization problem

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} Q(\mathbf{y}) = \arg\max_{\mathbf{y}} \mathbf{y}^\top\mathbf{A}(\mathbf{e}-\mathbf{y}).$$

Define a new function $f(\mathbf{y})$ as

$$f(\mathbf{y}) = \mathbf{y}^\top\mathbf{A}(\mathbf{e}-\mathbf{y}).$$

Again, the variable $\mathbf{y} \in \mathbb{B}^n$ is a binary vector and $\mathbf{e} = [1,1,\cdots,1]^\top$ is the unit column vector. Now we show that maximization of the above function $\max_{\mathbf{y}} f(\mathbf{y})$ is exactly a Max-Cut problem if we treat the symmetric matrix $\mathbf{A}$ as the weighted adjacency matrix of an undirected graph $\mathcal{G}_\mathbf{A} = \{V_\mathbf{A}, \mathbf{A}\}$. Note that the diagonal elements of $\mathbf{A}$ could be non-zero $\mathbf{A}_{ii} \neq 0, i = 1,2,\cdots,n$, which indicates the undirected graph $\mathcal{G}_\mathbf{A}$ has self-connected nodes. Assume $\mathbf{A} = \mathbf{A}^0 + \mathbf{A}^\Lambda$, where $\mathbf{A}^0$ is the matrix obtained by zeroing the diagonal elements of $\mathbf{A}$ and $\mathbf{A}^\Lambda$ is a diagonal matrix with $\mathbf{A}^\Lambda_{ii} = \mathbf{A}_{ii}, \mathbf{A}^\Lambda_{ij} = 0, i, j = 1,2,\cdots,n, i \neq j$. It is straightforward to show that the the function $f(\mathbf{y})$ can be written as

$$f(\mathbf{y}) = \mathbf{y}^\top(\mathbf{A}^0 + \mathbf{A}^\Lambda)(\mathbf{e}-\mathbf{y}) = \mathbf{y}^\top\mathbf{A}^0(\mathbf{e}-\mathbf{y}).$$

In other words, the non-zero elements in $\mathbf{A}$ do not affect the value of $f(\mathbf{y})$. Therefore, in the rest of this article, we can assume that the matrix $\mathbf{A}$ has zero diagonal elements unless the text specifies otherwise.

Since $\mathbf{y} = \{y_1, y_2, \cdots, y_n\}$ is a binary vector, each setting of $\mathbf{y}$ partitions the vertex set $V_\mathbf{A}$ in the graph $\mathcal{G}_\mathbf{A}$ into two disjoint subsets (S_1, S_2). In other words, the two subsets $S_1 = \{v_i | y_i = 1\}$ and $S_2 = \{v_i | y_i = 0\}$ satisfy $S_1 \cup S_2 = V_\mathbf{A}$ and $S_1 \cap S_2 = \emptyset$. The maximization problem can then be written as

$$\max f(\mathbf{y}) = \max \sum_{i,j} a_{ij} \cdot y_i(1 - y_j) = \max \frac{1}{2} \sum_{\substack{v_i \in S_1 \\ v_j \in S_2}} a_{ij}.$$

Because each binary vector $\mathbf{y}$ resulting in a partition (S_1, S_2) over the graph $\mathcal{G}_\mathbf{A}$ and $f(\mathbf{y})$ is a corresponding *cut*, the above maximization $\max f(\mathbf{y})$ is easily recognized as a Max-Cut problem (Deza and Laurent, 2009). However, in graph based semi-supervised learning, the variable $\mathbf{y}$ is partially specified by the initial label values. This given label information can be interpreted as a set of linear constraints on the Max-Cut problem. Thus, the optimal solution can achieved by solving a linearly constrained Max-Cut problem (Karp, 1972). In addition, we also show that a multi-class problem equals a Max K-Cut problem ($K = c$) (refer to Appendix A). Note that previous work used min-cut over the original data graph $\mathcal{G} = \{\mathbf{X}, \mathbf{W}\}$ to perform semi-supervised learning (Blum and Chawla, 2001; Blum et al., 2004). A key difference of the above formulation lies in the fact that we perform max-cut over the transformed graph $\mathcal{G}_\mathbf{A} = \{\mathbf{X}, \mathbf{A}\}$.

However, since there is no guarantee that the weights on the graph $\mathcal{G}_\mathbf{A}$ are non-negative, solutions to the Max-Cut problem can be difficult to find (Barahona et al., 1988). Therefore, in the following subsection, we will propose a gradient greedy solution to efficiently solve the above Max-Cut problem, which can be treated as a different view of the previous alternating minimization solution.

4.2 Label Propagation by Gradient Greedy Max-Cut

For the standard Max-Cut problem, many approximation techniques have been developed, including the most remarkable Goemans-Williamson algorithm using semidefinite programming (Goemans and Williamson, 1994, 1995). However, applying these guaranteed approximation schemes to solve the constrained Max-Cut problem for $\mathbf{Y}$ mentioned above is infeasible due to the constraints on initial labels. Furthermore, there is no guarantee that all edge weights a_{ij} of the graph $\mathcal{G}_\mathbf{A}$ are non-negative, a fundamental requirement in solving a standard Max-Cut problem (Goemans and Williamson, 1995). Instead, here we use a greedy gradient based strategy to find local optima by assigning each unlabeled vertex to the label set with minimum connectivity to maximize cross-set edge weights iteratively.

The greedy Max-Cut algorithm randomly selects unlabeled vertices and places each of them into the appropriate class subset depending on the edges between this unlabeled vertex and the vertices in the labeled subset. Given the label information, the initial label set for class j can be constructed as $\mathcal{S}_j = \{\mathbf{x}_i | y_{ij} = 1\}$ or $\mathcal{S}_j = \{\mathbf{x}_i | z_i = j\}$, $i = 1, 2, \cdots, n; j = 1, 2, \cdots, c$. Define the following as the connectivity between unlabeled vertex $\mathbf{x}_i$ and labeled subset $\mathcal{S}_j$

$$c_{ij} = \sum_{m=1}^{n} a_{im} y_{mj} = \mathbf{A}_{i.} \mathbf{Y}_{.j}, \tag{15}$$

Algorithm 1 Greedy Max-Cut for Label Propagation

Input: the graph $\mathcal{G}_{\mathbf{A}} = \{\mathbf{X}, \mathbf{A}\}$, the given labeled vertex $\mathbf{X}_l$, and initial labels $\mathbf{Y}$;
Initialization:
obtain the initial cut $\{\mathcal{S}_j\}$ by assigning the labeled vertex $\mathbf{X}_l$ to each subset:

$$\mathcal{S}_j = \{x_i | y_{ij} = 1\}, j = 1, 2, \cdots, c$$

unlabeled vertex set $\mathbf{X}_u = \mathbf{X} \setminus \mathbf{X}_l$;
repeat
randomly select an unlabeled vertex $\mathbf{x}_i \in \mathbf{X}_u$
compute the connectivity $c_{ij}, j = 1, 2, \cdots, c$
place the vertex to the labeled subject $\mathcal{S}_{j^*}$:

$$j^* = \arg\min_j c_{ij}$$

add $\mathbf{x}_i$ to $\mathbf{X}_l$: $\mathbf{X}_l \longleftarrow \mathbf{X}_l + \mathbf{x}_i$;
remove $\mathbf{x}_i$ from $\mathbf{X}_u$: $\mathbf{X}_u \longleftarrow \mathbf{X}_u - \mathbf{x}_i$;
until $\mathbf{X}_u = \emptyset$
Output: the final cut and the corresponding labeled subsets $\mathcal{S}_j, j = 1, 2, \cdots, c$

where $\mathbf{A}_{i\cdot}$ is the i'th row vector of $\mathbf{A}$ and $\mathbf{Y}_{\cdot j}$ is the j'th column vector of $\mathbf{Y}$. Intuitively, c_{ij} represents the sum of edge weights between vertex $\mathbf{x}_i$ and label set $\mathcal{S}_j$ given the graph $\mathcal{G}_{\mathbf{A}}$ with edge weights $\mathbf{A}$. Based on this definition, a straightforward local search for the maximum cut involves placing each unlabeled vertex $\mathbf{x}_i \in \mathbf{X}_u$ in the labeled subset $\mathcal{S}_j$ with minimum connectivity c_{ij} to maximize the cross-set edge weights as shown in Algorithm (1). In order to achieve a good solution Algorithm (1) should be run multiple times with different random seeds after which the best cut overall is output (Mathieu and Schudy, 2008).

While the above method is computationally cumbersome, it still does not resolve the issue of undesired local optima and may generate biased cuts. According to the definition in Equation (15), the initialized labels determine the connectivity between unlabeled vertices and labeled subsets. If the computed connectivity is negative, the above random search will prefer assigning unlabeled vertices to the label set with the most labeled vertices which results in biased partitioning. Such biased partitioning also occurs in minimum cut problems over an undirected graph with positive weights (Shi and Malik, 2000). Other label initialization problems may also produce a poor cut. For example, the numbers of labels from different classes may deviate from the underlying class proportions. Alternatively, the labeled vertices may be outliers and lie in regions of the graph with low density. Such labels often lead to weak label prediction results (Wang et al., 2008a). Furthermore, the algorithm's random selection of an unlabeled vertex results in unstable predictions since the chosen unlabeled vertex $\mathbf{x}_i$ could have equally low connectivity to multiple label subsets $\mathcal{S}_j$.

To address the aforementioned issues, we first modify the original definition of connectivity to alleviate label imbalance across different classes. A weighted connectivity is computed as

$$c_{ij} = p_j \cdot \sum_{m=1}^{n} \lambda_m a_{im} y_{mj} = p_j \cdot \mathbf{\Lambda} \mathbf{A}_{i\cdot} \mathbf{Y}_{\cdot j}. \tag{16}$$

The diagonal matrix $\mathbf{\Lambda} = \text{diag}([\lambda_1, \lambda_2, \cdots, \lambda_n]$ is called the label weight matrix as in Equation (10)

$$\lambda_i = \begin{cases} d_i / d_{\mathcal{S}_j} & : \text{ if } \mathbf{x}_i \in \mathcal{S}_j, j = 1, \cdots, c \\ 0 & : \text{ otherwise,} \end{cases}$$

Algorithm 2 Greedy Gradient based Max-Cut for Label Propagation

Input: the graph $\mathcal{G}_{\mathbf{A}} = \{\mathbf{X}, \mathbf{A}\}$ and the given labeled vertex $\mathbf{X}_l$, and initial label $\mathbf{Y}$;
Initialization:
obtain the initial cut $\{\mathcal{S}_j\}$ through assigning the labeled vertex $\mathbf{X}_l$ to each subset:

$$\mathcal{S}_j = \{x_i | y_{ij} = 1\}, j = 1, 2, \cdots, c$$

unlabeled vertex set $\mathbf{X}_u = \mathbf{X} \setminus \mathbf{X}_l$;
repeat
  for all $j = 0$ to $|\mathbf{X}_u|$ **do**
    compute weighted connectivity:

$$c_{ij} = \sum_{k=1}^{n} \lambda_i a_{ik} y_{kj}, \mathbf{x}_i \in \mathbf{X}_u, j = 1, \cdots, c$$

  end for
  update the cut $\{\mathcal{S}_i\}$ by placing the vertex $\mathbf{x}_{i^*}$ to the $\mathcal{S}_{j^*}$'th subset:

$$(i^*, j^*) = \arg \min_{i,j,\mathbf{x}_i \in \mathbf{X}_u} c_{ij}$$

  add $\mathbf{x}_i$ to $\mathbf{X}_l$: $\mathbf{X}_l \longleftarrow \mathbf{X}_l + \mathbf{x}_i$;
  remove $\mathbf{x}_i$ from $\mathbf{X}_u$: $\mathbf{X}_u \longleftarrow \mathbf{X}_u - \mathbf{x}_i$;
until $\mathbf{X}_u = \emptyset$
Output: the final cut and the corresponding labeled subsets $\mathcal{S}_j, j = 1, 2, \cdots, c$

where $d_{\mathcal{S}_j} = \sum_{\mathbf{x}_m \in \mathcal{S}_j} d_m$ is the sum of the degrees of the vertices in the label set $\mathcal{S}_j$. This heuristic setting weights the importance of each label based on degree which alleviates the adverse impact of outliers. If we ignore class priors, this definition of $\mathbf{\Lambda}$ coincides with the one in Equation (10).

Finally, to handle any instability due to the random search algorithm, we propose a greedy gradient search approach where the most beneficial vertex is assigned to the label set with minimum connectivity. In other words, we first compute the connectivity matrix $\mathbf{C} = \{c_{ij}\} \in \mathbb{R}^{n \times c}$ that gives the connectivity between all unlabeled vertices to existing label sets

$$\mathbf{C} = \mathbf{A}\mathbf{\Lambda}\mathbf{Y}.$$

Then we examine $\mathbf{C}$ to identify the element (i^*, j^*) with minimum value as

$$(i^*, j^*) = \arg \min_{i,j:\mathbf{x}_i \in \mathbf{X}_u} c_{ij}.$$

This means that the unlabeled vertex $\mathbf{x}_{i^*}$ has the least connectivity with label set $\mathcal{S}_{j^*}$. Then, we update the labeled set $\mathcal{S}_{j^*}$ by adding vertex $\mathbf{x}_{i^*}$ as one greedy step to maximize the cross-set edge weights. This greedy search can be repeated until all the unlabeled vertices are assigned to labeled sets. In each iteration of the greedy cut process, the weighted connectivity of all unlabeled vertices to labeled sets is re-computed. Then the vertex with minimum connectivity is placed in the proper labeled set. The algorithm is summarized in Algorithm (2).

The connectivity matrix $\mathbf{C}$ can also be viewed as the gradient of the cost function Q in Equation (12) with respect to $\tilde{\mathbf{Y}}$. This is precisely the same setting used in Equation (13) of the alternating minimization algorithm

$$\mathbf{C} = \frac{\partial Q}{\partial \tilde{\mathbf{Y}}} = \mathbf{A}\mathbf{\Lambda}\mathbf{Y}.$$

We name the algorithm *greedy gradient Max-Cut* (*GGMC*) since, in the greedy step, the unlabeled vertices are assigned labels in a manner that reduces the value of Q along the direction of the steepest descent. Consider both the variables $\mathbf{Y}$ and $\mathbf{F}$ in the original bivariate formulation in Equation (9). The greedy Max-Cut method is equivalent to the alternating minimization procedure discussed earlier. Unlike graph-cut based *SSL* methods such as mincuts (Blum and Chawla, 2001; Blum et al., 2004), our *GGMC* algorithm tends to generate more natural graph cuts and avoid biased solutions since it uses a weighted connectivity matrix. This allows it to effectively handle the issues mentioned earlier and, in practice, achieve significant gains in accuracy while retaining efficiency.

4.3 Complexity and Speed Up

Assume the graph has $n = |\mathbf{X}|$ vertices and a subset $\mathbf{X}_l$ with $l = |\mathbf{X}_l|$ labeled vertices (where $l \ll n$). The greedy gradient algorithm terminates after at most $n - l \simeq n$ iterations. In each iteration of the greedy gradient algorithm, the connectivity matrix $\mathbf{C}$ is updated by a matrix multiplication (an $n \times n$-matrix is multiplied by a $n \times c$-matrix). Hence, the complexity of the greedy algorithm is $O(cn^3)$.

However, the greedy algorithm can be greatly accelerated in practice. For example, the computation of the connectivity in Equation (16) can be done incrementally after assigning each new unlabeled vertex to a certain label set. This circumvents the re-calculation of all the entries in the $\mathbf{C}$ matrix. Assume in the t'th iteration the connectivity is $\mathbf{C}^t$ and an unlabeled vertex $\mathbf{x}_i$ with degree d_i is assigned to the labeled set $\mathcal{S}_j$. Clearly, for all remaining unlabeled vertices, the connectivity to the labeled sets remains unchanged except for the j'th labeled set. In other words, only the j'th column of $\mathbf{C}$ needs updating. This update is performed incrementally via

$$\mathbf{C}_{.j}^{t+1} = \frac{d_{\mathcal{S}_j^t}}{d_{\mathcal{S}_j^{t+1}}} \mathbf{C}_{.j}^t + \frac{d_i}{d_{\mathcal{S}_j^{t+1}}} \mathbf{A}_{.i},$$

where $d_{\mathcal{S}_j^{t+1}} = d_{\mathcal{S}_j^t} + d_i$ is the sum of the degrees of the labeled vertices after assigning $\mathbf{x}_i$ to the labeled set $\mathcal{S}_j$. This incremental update reduces the complexity of the greedy gradient search algorithm to $O(n^2)$.

5. Experiments

In this section, we demonstrate the superiority of the proposed *GGMC* method over state-of-the-art semi-supervised learning methods using both synthetic and real data. Previous work showed that *LapSVM* and *LapRLS* outperform other semi-supervised approaches such as Transductive SVMs *TSVM* (Joachims, 1999) and ∇*TSVM* (Chapelle and Zien, 2005). Therefore, we limit our comparisons to only the *LapRLS*, *LapSVM* (Sindhwani et al., 2005; Belkin et al., 2005), *LGC* (Zhou et al., 2004) and *GFHF* (Zhu et al., 2003). To set various hyper-parameters such as γ_I, γ_r in *LapRLS* and *LapSVM*, we followed the default configurations used in the literature. Similarly, for *GGMC* and *LGC*, we set the hyper-parameter $\mu = 0.01$ across all data sets. For the computational cost, *GGMC*, *LGC* and *GFHF* required very similar run-times to output a prediction. However, *LapRLS* and *LapSVM* need significant longer training time, especially for multiple-class problems since multiple rounds of one-vs-all training have to be performed.

As in Section 2, any real implementation of graph-based *SSL* needs a graph construction method algorithm that builds a graph from the training data $\mathbf{X}$. This is then followed by a sparsification

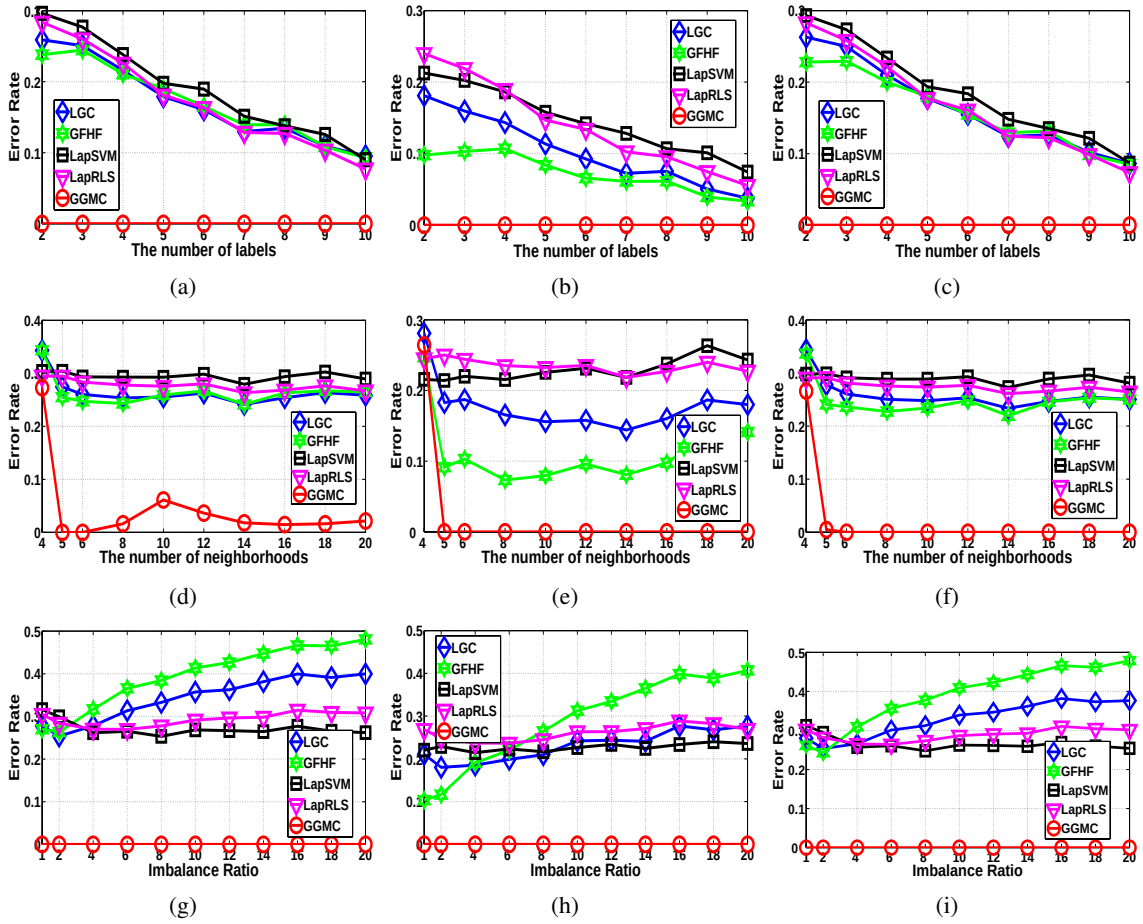
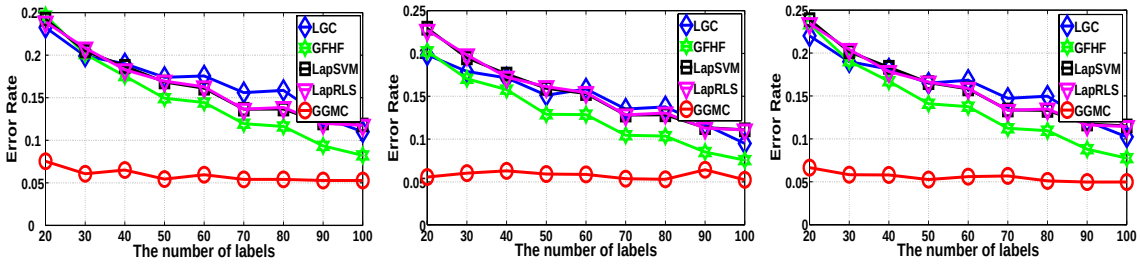


Figure 4: Experimental results on the noisy two-moon data set simulating different graph construction approaches and label conditions. Figures a) d) g) use binary weighting. Figures b) e) h) use fixed Gaussian kernel weighting. Figures c) f) i) use adaptive Gaussian kernel weighting. Figures a) b) c) vary the number of labels. Figures d) e) f) vary the value of k in the graph construction. Figures g) h) i) vary the label imbalance ratio.

procedure to generate the sparse connectivity matrix $\mathbf{B}$ and a weighting procedure to obtain weights on the edges in $\mathbf{B}$. In these experiments, we used the same graph construction procedure for all the *SSL* algorithms. The sparsification was done using the standard k-nearest-neighbors approach and the edge weighting involved either *binary weighting* or *Gaussian kernel weighting*. In the latter case, the ℓ_2 distance $d_{\ell_2}(\mathbf{x}_i, \mathbf{x}_j)$ is used and the kernel bandwidth σ is estimated in two different ways. The first estimate uses a fixed σ defined as the average distance between each selected sample and its k'th nearest neighbor (Chapelle et al., 2006). In addition, a second adaptive approach is also considered which locally estimates the parameter σ to the mean distance in the k-nearest neighborhoods of the samples $\mathbf{x}_i$ and $\mathbf{x}_j$ (Wang et al., 2008a).



(a) (b) (c)

Figure 5: Experimental results on the USPS digits data set under varying levels of labeling using: a) binary weighting; b) fixed Gaussian kernel weighting and c) adaptive Gaussian kernel weighting.

5.1 Noisy Two-Moon Data Set

We first compared *GGMC* with several representative *SSL* algorithms using the noisy two-moon data set shown in Figure 3. Despite the near-perfect classification results reported on clean versions of this data set (Sindhwani et al., 2005; Zhou et al., 2004), small amounts of noise quickly degrade the performance of previous algorithms. In Figure 3, two separable manifolds containing 600 two-dimensional points are mixed with 100 noisy outlier samples. The noise foils previous methods which are sensitive to the locations of the initial labels, disproportional sampling from the classes, and outlier noise. All experiments are repeated with 100 independent folds with random sampling to show the average error rate of each algorithm.

The first group of experiments varies the number of labels provided to the algorithms. We uniformly used $k = 6$ in the k-nearest-neighbors graph construction and applied the aforementioned three edge-weighting schemes. The average error rates of the predictions on the unlabeled points is shown in Figures 4(a), 4(b) and 4(c). These correspond to binary edge weighting, fixed Gaussian kernel edge weighting, and adaptive Gaussian kernel edge weighting, respectively. The results clearly show that *GGMC* is robust to the size of the label set and and generates perfect prediction results for all three edge-weighting schemes.

The second group of experiments demonstrate the influence of the number of edges (i.e., the value of k) in the graph construction method. We varied the value of k from 4 to 20 and Figures 4(d), 4(e), and 4(f) show results for the different edge-weighting schemes. Once again, *GGMC* achieves significantly better performance in most cases.

Finally, we studied the effect of imbalanced labeling on the *SSL* algorithms. We fix one class to have only one label and then randomly select r labels from the other classes. Here, r indicates the imbalance ratio and we study the range $1 \leq r \leq 20$. Figures 4(g), 4(h), and 4(i) show the results with different edge-weighting schemes. Clearly, *GGMC* is insensitive to the imbalance since it computes a per-class label weight normalization which compensates directly for differences in label proportions.

In summary, Figure 4 depicted the performance advantages of *GGMC* relative to *LGC*, *GFHF*, *LapRLS*, and *LapSVM* methods. We clearly see that the four previous algorithms are sensitive to the initial labeling conditions and none of them produces perfect prediction. Furthermore, the error rates of *LGC* and *GFHF* increase significantly when labeling becomes imbalanced, even if many

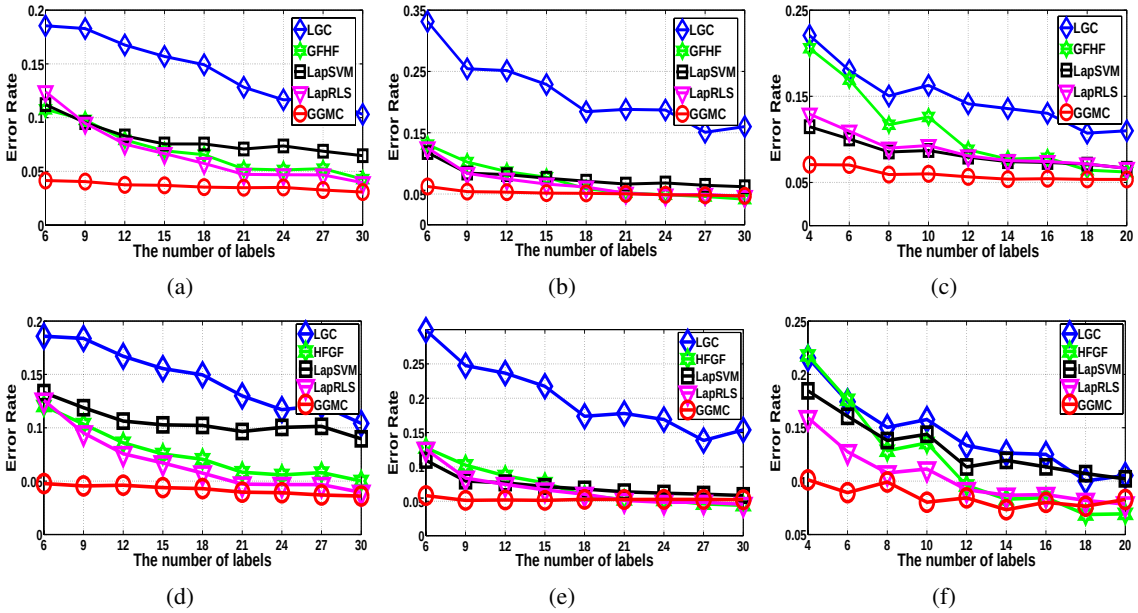


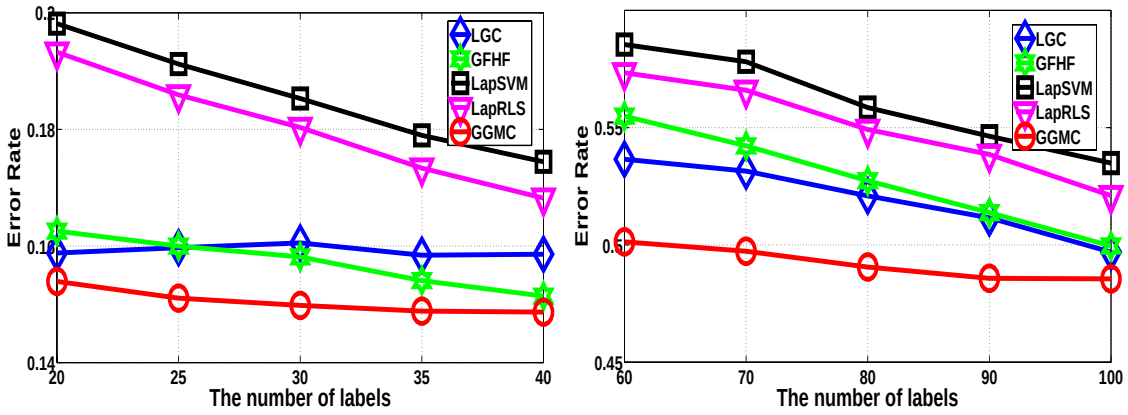
Figure 6: Performance of *LGC*, *GFHF*, *LapRLS*, *LapSVM*, and *GGMC* algorithms using the UCI data sets. The horizontal axis is the number of training labels provided while the vertical axis is the average error rate achieved over 100 random folds. Results are based on k-nearest neighbor graphs shown in a) for the Iris data set, in b) for the Wine data set and in c) for the Breast Cancer data set. Results are based on b-matched graphs shown in d) for the Iris data set, in e) for the Wine data set and in f) for the Breast Cancer data set.

labels are made available. However, *GGMC* achieves high accuracy regardless of the imbalance ratio and the size of the label set. Furthermore, *GGMC* remains robust to the graph construction procedure and the edge-weighting strategy.

5.2 Handwritten Digit Data Set

We also evaluated the algorithms in an image recognition task where handwritten digits in the USPS database are to be annotated with the appropriate label $\{0,1,\ldots,9\}$. The data set contains gray scale handwritten digit images involving 16×16 pixels. We randomly sampled a subset of 4000 samples from the data. For all the constructed graphs, we used the k-nearest-neighbors algorithm with $k=6$ and tried the three different edge-weighting schemes above. We varied the total number of labels from 20 to 100 while guaranteeing that each digit class had at least one label. For each setting, the average error rate was computed over 20 random folds.

The experimental results are shown in Figures 5(a), 5(b), and 5(c) which correspond to the three different edge-weighting schemes. As usual, *GGMC* significantly improves classification accuracy relative to other approaches, especially when few labeled training examples are available. The average error rates of *GGMC* are consistently low with small standard deviations. This demonstrates that the *GGMC* method is less sensitive to the number and locations of the initial training labels.



(a) (b)

Figure 7: Performance of *LGC*, *GFHF*, *LapRLS*, *LapSVM*, and *GGMC* algorithms using the COIL-20 and Animal data sets. The horizontal axis is the number of training labels provided while the vertical axis is the average error rate. Results are shown in a) for the COIL-210 object data set, and in b) for the Animal data set.

5.3 UCI Data Sets

We tested *GGMC* and the other algorithms on benchmark data sets from the UCI Machine Learning Repository (Frank and Asuncion, 2010). Specifically, we used the Iris, Wine, and Breast Cancer data sets. The numerical attributes of the data sets are all normalized to span the range $[0,1]$. For all three data sets, we used a k-nearest-neighbors graph construction procedure with $k = 6$ and explored the Gaussian kernel with fixed bandwidth as the edge-weighting scheme, where the bandwidth is set as the average distance between each selected sample and its k'th nearest neighbor.

Figure 6 shows the performance of the various *SSL* algorithms. The vertical axis is the average error rate computed over 100 random folds and the horizontal axis shows the number of labeled samples provided at training time. Besides using the k-nearest neighbor graphs, we also evaluated the perform using the b-matched graphs on this data set. The *GGMC* method significantly outperforms other algorithms in most test cases, especially when little labeled data is available.

5.4 COIL-20 Object Images

We investigated the object recognition problem using the well-known Columbia Object Image Library (COIL-20), which contains 1440 gray-scale images of 20 objects (Nene et al., 1996). The images sequences were obtained when the objects were placed on a turntable table with black background, where one image was taken for each 5-degree interval. As with UCI data sets, we constructed kNN graphs with $k = 6$ and used a fixed bandwidth for edge weighting. The number of given labels from all object categories was varied from 20 to 40 with the guarantee that each object class has at least one label. Figure 7(a) shows the performance curves in terms of the average error rate of 100 random tests, where *GGMC* outperformed all other methods.

5.5 NEC Animal Data Set

The NEC Animal data set contains sequences of images of 60 toy animals and has been used as a benchmark data set for image and video classification (Mobahi et al., 2009). Each toy animal has around 72 images taken at different poses. The data set contains a total of 4371 images, each of size 580×480 pixels. In the experiments, the images were re-sized to 96×72 pixels and the grey intensity was used as the feature representation. The previous graph construction methodology was followed and algorithm performance was evaluated using the average error rate across 100 random folds with the number of initial labels varying from 60 to 100. In the experiments, the *GGMC* method again achieved the best performance among all tested methods. In particular, when given very sparse labels, that is, one label per class, *GGMC* produced significantly lower error rates.

6. Conclusion and Discussion

The performance of existing graph-based *SSL* methods depends heavily on the availability of accurate initial labels and good connectivity structure. Otherwise, performance can significantly degrade if labels are not distributed evenly across classes, if the initial label locations are biased, or if noise and outliers corrupt the underlying manifold structure. These problems arise in many real world data sets and limit the performance of state-of-the-art *SSL* algorithms. Furthermore, several heuristic choices in the *SSL* approach require considerable exploratory work by the practitioner before the methods perform well in a specific problem domain.

This article addressed these shortcomings and proposed a novel graph-based semi-supervised learning method named greedy gradient Max-Cut (*GGMC*). Our main contributions include:

1. Extending the existing univariate quadratic regularization framework to an optimization over both label matrix and classification function. Such an extension allows us to treat input labels as part of the optimization problem and thereby alleviate *SSL*'s sensitivity to initial labels.

2. Demonstrating that the bivariate formulation is actually a mixed integer programming problem which can be reduced to a binary integer programming (BIP) problem. In addition, we show that an alternating minimization procedure can be used to derive a locally optimal solution.

3. Proving that the proposed bivariate formulation is equivalent to a Max-Cut problem for the two-class case and proving that it is equivalent to a Maximum K-cut problem for the multi-class case. In addition, we proposed an efficient solution with $O(n^2)$ complexity. This greedy gradient Max-Cut (*GGMC*) solution presents a different interpretation for the alternating minimization procedure from a graph cut view.

Unlike other graph-cut based *SSL* methods such as min-cut (Blum and Chawla, 2001; Blum et al., 2004), the proposed *GGMC* algorithm tends to generate more natural graph cuts and avoids biased solutions. In addition, it uses a weighted connectivity matrix to normalize the label matrix. The result is a solution that can cope with all the aforementioned degeneracies. It improves accuracy in practice while remaining efficient. Future work will extend the proposed methods to out-of-sample settings where additional data points are added to the prediction problem without requiring a full retraining procedure. Another interesting extension of the bivariate framework is active learning which can potentially reduce the amount of labels necessary for accurate prediction (Goldberg et al., 2011).

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1117631. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation. This work was also partially supported by a Google Research Award and by the Department of Homeland Security under Grant No. N66001-09-C-0080.

Appendix A. Multi-Class Case as a Max K-Cut Problem

Here, we show that K-class bivariate graph transduction is equivalent to a Max K-Cut problem. If the number of classes is K, the label variable $\mathbf{Y}$ is a $n \times K$ matrix denoting the classification result. Therein, $\mathbf{Y}_{ij} = 1$ indicates that vertex $\mathbf{x}_i$ is assigned the label j. We rewrite the cost function in Equation (8) as

$$Q(\mathbf{Y}) = \frac{1}{2}\mathrm{tr}\left(\mathbf{Y}^\top \mathbf{A}\mathbf{Y}\right) = \frac{1}{2}\sum_{k=1}^{K} \mathbf{Y}_{.k}^\top \mathbf{A}\mathbf{Y}_{.k}.$$

Let $\mathbf{y}_k = \mathbf{Y}_{.k}$ be a column vector of $\mathbf{Y}$. Let the non-zero elements in $\mathbf{y}_k$ denote the vertices in subset S_k, where $k = 1, 2, \cdots, K$, $S_1 \cup S_2 \cup \cdots \cup S_K = V_\mathbf{A}$, and $S_m \cap S_n = \emptyset$ if $m \neq n$. Then the above cost function is equivalent to

$$\begin{aligned} Q(\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_K) &= \tfrac{1}{2}\sum_{k=1}^{K} \mathbf{y}_k^\top \mathbf{A}\mathbf{y}_k = \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in S_k \\ i<j}} \mathbf{A}_{ij} \\ &= \sum_{i<j} \mathbf{A}_{ij} - \sum_{m=1}^{K-1} \sum_{n=m+1}^{K} \sum_{\substack{\mathbf{x}_i \in S_m \\ \mathbf{x}_j \in S_n}} \mathbf{A}_{ij}. \end{aligned}$$

Therefore the original minimization problem is equivalent to maximizing the sum of the weight of the edges between the disjoint sets S_k, that is, the maximum K-cut problem

$$\max_{S_1,\ldots,S_K} \sum_{m=1}^{K-1} \sum_{n=m+1}^{K} \sum_{\substack{\mathbf{x}_i \in S_m \\ \mathbf{x}_j \in S_n}} \mathbf{A}_{ij}.$$

References

A. Azran. The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In *Proceedings of the International Conference on Machine Learning*, pages 49–56, Corvalis, Oregon, 2007. ACM.

M.-F. Balcan, A. Blum, and K. Yang. Co-training and expansion: towards bridging theory and practice. In *Advances in Neural Information Processing Systems*, volume 17, pages 89–96. 2005.

F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988.

M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, Barbados, January 2005.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: a Geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

T. D. Bie and N. Cristianini. Convex methods for transduction. In *Advances in Neural Information Processing Systems*, volume 16. 2004.

A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of International Conference on Machine Learning*, pages 19–26, San Francisco, CA, USA, 2001.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100, Madison, Wisconsin, United States, 1998. ACM.

A. Blum, J. Lafferty, M. R. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *Proceedings of the Twenty-first International Conference on Machine Learning*, pages 13–20, Banff, Alberta, Canada, 2004.

M. Carreira-Perpinán and R. S. Zemel. Proximity graphs for clustering and manifold learning. In *Advances in neural information processing systems*, volume 17, pages 225–232. 2005.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, Barbados, January 2005.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. URL `http://www.kyb.tuebingen.mpg.de/ssl-book`.

O. Chapelle, V. Sindhwani, and S. S. Keerthi. Branch and bound for semi-supervised support vector machines. In *Advances in Neural Information Processing Systems*, volume 19, pages 217–224. Cambridge, MA, 2007.

O. Chapelle, V. Sindhwani, and S. S. Keerthi. Optimization techniques for semi-supervised support vector machines. *The Journal of Machine Learning Research*, 9:203–233, 2008.

N. V. Chawla and G. Karakoulas. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research*, 23(1):331–366, 2005.

F. R. K. Chung and N. Biggs. *Spectral Graph Theory*. American Mathematical Society Providence, RI, 1997.

S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio, United States, 1971. ACM.

M. M. Deza and M. Laurent. *Geometry of Cuts and Metrics*. Springer Verlag, 2009.

J. Edmonds and E. Johnson. Matching: A well-solved class of integer linear programs. *Combinatorial OptimizationEureka, You Shrink!*, pages 27–30, 2003.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

M. X. Goemans and D. P. Williamson. .879-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, pages 422–431, Montreal, Quebec, Canada, 1994.

M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

A.B. Goldberg, X. Zhu, A. Furger, and J.M. Xu. Oasis: Online active semi-supervised learning. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

S. Goldman and Y. Zhou. Enhancing supervised learning with unlabeled data. In *Proceedings of the 17th International Conference on Machine Learning*, pages 327–334, 2000.

B. Huang and T. Jebara. Loopy belief propagation for bipartite maximum weight b-matching. In *Int. Workshop on Artificial Intelligence and Statistics*, 2007.

B. Huang and T. Jebara. Collaborative filtering via rating concentration. In Y.W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume Volume 9 of JMLR: W&CP, pages 334–341, May 13-15 2010.

T. Jebara, J. Wang, and S.-F. Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 441–448, 2009.

T. Joachims. Transductive inference for text classifcation using support vector machines. In *Proceedings of International Conference on Machine Learning*, pages 200–209, 1999.

T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of International Conference on Machine Learning*, pages 290–297, 2003.

R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 43:85–103, 1972.

M.-A. Krogel and T. Scheffer. Multi-relational learning, text mining, and semi-supervised learning for functional genomics. *Machine Learning*, 57(1):61–81, 2004.

B. Kveton, M. Valko, A. Rahimi, and L. Huang. Semi-supervised learning with max-margin graph cuts. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 421–428, 2010.

W. Liu, J. Wang, and S.-F. Chang. Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, 100(9):2624–2638, 2012.

M. Maier, U. von Luxburg, and M. Hein. Influence of graph construction on graph-based clustering measures. In *Advances in Neural Information Processing Systems*, volume 22, pages 1025–1032. 2009.

G. S. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of International Conference on Machine Learning*, pages 593–600, Corvalis, Oregon, 2007.

C. Mathieu and W. Schudy. Yet another algorithm for dense max cut: go greedy. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 176–182, 2008.

S. Melacci and M. Belkin. Laplacian support vector machines trained in the primal. *Journal of Machine Learning Research*, 12:1149–1184, 2011.

H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744, 2009.

S.A. Nene, S.K. Nayar, and H. Murase. Columbia object image library (coil-20). *Dept. Comput. Sci., Columbia Univ., New York.[Online] http://www. cs. columbia. edu/CAVE/coil-20.html*, 1996.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 824–831, Bonn, Germany, 2005.

V. Sindhwani, J. Hu, and A. Mojsilovic. Regularized co-clustering with dual supervision. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 976–983. MIT Press, 2008.

M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, pages 945–952. 2002.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

J. Wang, S.-F. Chang, X. Zhou, and T. C. S. Wong. Active Microscopic Cellular Image Annotation by Superposable Graph Transduction with Imbalanced Labels. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Alaska, USA, June 2008a.

J. Wang, T. Jebara, and S.-F. Chang. Graph transduction via alternating minimization. In *Proceedings of International Conference on Machine Learning*, pages 1144–1151, Helsinki, Finland, 2008b.

J. Wang, Y.-G. Jiang, and S.-F. Chang. Label diagnosis through self tuning for web image search. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Miami Beach, Florida, USA, June 2009.

W. Wang and Z.-H. Zhou. A new analysis of co-training. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1135–1142, Haifa, Israel, June 2010.

Z. Xu, R. Jin, J. Zhu, I. King, and M. Lyu. Efficient convex relaxation for transductive support vector machine. volume 21, pages 1641–1648. 2008.

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16, pages 321–328. 2004.

X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

X. Zhu and A. B. Goldberg. *Introduction to Semi-supervised Learning*. Morgan & Claypool Publishers, 2009.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of International Conference on Machine Learning*, pages 912–919, 2003.