Graph Construction and b-Matching for Semi-Supervised Learning

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

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

Abstract

Graph based semi-supervised learning (SSL) methods play an increasingly important role in practical machine learning systems. A crucial step in graph based SSL methods is the conversion of data into a weighted graph. However, most of the SSL literature focuses on developing label inference algorithms without extensively studying the graph building method and its effect on performance. This article provides an empirical study of leading semi-supervised methods under a wide range of graph construction algorithms. These SSL inference algorithms include the Local and Global Consistency (LGC) method, the Gaussian Random Field (GRF) method, the Graph Transduction via Alternating Minimization ($GTAM$) method as well as other techniques. Several approaches for graph construction, sparsification and weighting are explored including the popular k-nearest neighbors method (kNN) and the b-matching method. As opposed to the greedily constructed kNN graph, the b-matched graph ensures each node in the graph has the same number of edges and produces a balanced or regular graph. Experimental results on both artificial data and real benchmark datasets indicate that b-matching produces more robust graphs and therefore provides significantly better prediction accuracy without any significant change in computation time.

Appearing in *Proceedings of the 26^{th} International Conference on Machine Learning*, Montreal, Canada, 2009.

1. Introduction

Recently, semi-supervised learning (SSL) has proliferated in applied machine learning settings since labeled data is often easily complemented with large unlabeled datasets. Of the current SSL methods, graph based approaches have emerged as methods of choice for general[1] semi-supervised tasks in terms of accuracy and computational efficiency. Graph based methods treat both labeled and unlabeled samples from the dataset as nodes in a graph and then instantiate pairwise edges between these nodes which are weighted by the affinity between the corresponding pairs of samples. The small portion of vertices with labels are then used by SSL methods to perform propagation or diffusion on the graph which provides unlabeled nodes with predicted labels. In many graph based SSL methods, these node labeling procedures are formalized as regularized function estimation on an undirected graph.

Many graph based SSL techniques involve three important choices. First, the user chooses a similarity function or kernel for estimating the affinity between pairs of samples or the edge weight between pairs of nodes. For both empirical and theoretical reasons (Belkin & Niyogi, 2008), the Gaussian kernel is widely used in most situations. Alternatively, for histogram data such as word counts or the TF-IDF representation of documents, the cosine measure or χ^2 distance may be preferred. Second, the user selects an algorithm for finding a sparse weighted subgraph from the fully connected weighted graph between all pairs of nodes. Sparsity is important to ensure that the SSL algorithms remain efficient and robust to noise. The algorithm that recovers the subgraph may prune some edges as well as reweight other

[1] In general (agnostic) settings when no parametric information is available about the data distribution, graph based semi-supervised learning performs well. If additional parametric information is known, other methods may be useful, however.

edges. The most common algorithm for recovering a sparse subgraph is the k nearest neighbors algorithm (kNN). Each node merely recovers its k neighbors using the similarity function and instantiates k undirected edges between itself and the neighbors. Another contender approach is the ϵ-neighborhood graph which merely includes edges if samples are within distance of ϵ away from each other. In the SSL literature (Zhu, 2005), the vertex connection, graph sparsification and edge weighting methods need to be explored to produce graphs that are appropriate for the subsequent label inference algorithms. The final step requires the user to select a graph based SSL algorithm, i.e. specify how to diffuse the labels on the known part of the graph to the unknown nodes. Most graph based SSL methods estimate a continuous classification function $F \in \mathbb{R}^{|V| \times c}$, where $|V|$ is the number of vertices and c is the number of label class, on the graph by optimizing a predefined energy function. Typically, the energy function imposes a trade-off between the empirical risk on labeled vertices and consistency or smoothness of the function on the graph (agreement in the predictions on closely connected vertices). The current best graph based SSL techniques include the min-cut method (Blum & Chawla, 2001), the Gaussian fields and harmonic functions method (Zhu et al., 2003), the local and global consistency method (Zhou et al., 2004), the manifold regularization or Laplacian support vector machine (Belkin et al., 2005) and, most recently, the alternating graph transduction method (Wang et al., 2008).

This paper thoroughly explores the various combinations of the above graph construction and label inference algorithms. In addition, a novel graph construction method is proposed which is known as b-matching as an alternative to the k nearest neighbor graph. Unlike k nearest neighbors which greedily adds the k closest points to each node and may return graphs where some nodes have many more than k neighbors (some nodes get selected very often by other nodes), b-matching ensures the graph is exactly regular: every node has b neighbors at termination. While b-matching is a popular tool in auctions, operations research, circuit layout and many other optimization settings, it has yet to be applied in semi-supervised settings. In addition it can be computationally demanding in traditional implementations. However, in the recent work (Huang & Jebara, 2007), a fast implementation of b-matching was developed using a belief propagation algorithm which (despite loops in the resulting Bayesian network) guarantees convergence to the global solution in cubic time in the number of nodes in the graph. Recent results show that (under some conditions) the belief propagation approach can converge in only quadratic time (Salez & Shah, 2009) for dense networks (further speedup is possible if the network itself is sparse to begin with). In practice, b-matching (using belief propagation) is not significantly more demanding than the label inference and label propagation methods that follow it. Therefore it is a suitable alternative to the k-nearest neighbor graph construction method. In investigating the graph construction aspect of SSL, this article also considers b-matching which, surprisingly, performs significantly better than kNN, the previous contender. While kNN is computationally efficient and leads to accurate SSL, greedily connecting neighborhood vertices usually results in imbalance between the degrees of vertices and uneven graph connectivity. This article proposes b-matching as a tool for graph based semi-supervised learning and experimental results on both artificial data and real benchmark data shows that it produces significant improvement in accuracy.

The remainder of this paper is organized as follows. In Section 2, we describe the details of the graph construction methods prior to the label inference stage, in particular neighborhood based methods and b-matching methods. Section 3 reviews various state-of-the art SSL algorithms. Section 4 provides experimental validation for the algorithms on both toy and benchmark datasets, including text classification and digit recognition. Comparisons with leading semi-supervised methods are investigated. Concluding remarks and a discussion are then provided in Section 5. Appendix A provides details of the implementation for b-matching based on belief propagation.

2. Graph Construction for Semi-Supervised Learning

Assume we are given *iid* (independent and identically distributed) labeled samples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$ as well as unlabeled samples $\{\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}\}$ drawn from a distribution $p(\mathbf{x}, y)$. The goal of semi-supervised learning is to infer the missing labels $\{y_{l+1}, \ldots, y_{l+u}\}$ corresponding to the unlabeled samples. Define the set of labeled inputs as $\mathcal{X}_l = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ with cardinality $|\mathcal{X}_l| = l$ and the set of unlabeled inputs $\mathcal{X}_u = \{\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}\}$ of cardinality $|\mathcal{X}_u| = u$. A crucial component of applying graph based semi-supervised learning is the estimation of a weighted undirected sparse graph $\mathcal{G}$ from the input data $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$. Subsequently, a labeling algorithm uses $\mathcal{G}$ and the known labels $\mathcal{Y}_l = \{y_1, \ldots, y_l\}$ to provide estimates $\hat{\mathcal{Y}}_u = \{\hat{y}_{l+1}, \ldots, \hat{y}_{l+u}\}$ which hopefully agree with the true labels $\mathcal{Y}_u = \{y_{l+1}, \ldots, y_{l+u}\}$ as measured by an

appropriately chosen loss function.

In this section, the graph construction $\mathcal{X} \rightarrow \mathcal{G}$ is addressed. In the subsequent section, the inference method $\{\mathcal{G}, \mathcal{Y}_l\} \rightarrow \mathcal{Y}_u$ is discussed. Given input data $\mathcal{X}$ of cardinality $|\mathcal{X}| = l + u$, graph construction produces a graph $\mathcal{G} = (V, E)$ consisting of $n = l + u$ vertices V where each vertex V_i is associated with the sample $\mathbf{x}_i$. Furthermore, take E to be the set of undirected edges between pairs of vertices. It is common to also associate a weighted symmetric adjacency matrix W with the edges E in $\mathcal{G}$ where $W \in \mathbb{R}^{n \times n}$ has zeros on its diagonal and each scalar W_{ij} represents the edge weight between node V_i and node V_j. The estimation of $\mathcal{G}$ from $\mathcal{X}$ usually proceeds in two steps.

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

2.1. Graph Sparsification

Starting with the fully connected matrix $A \in \mathbb{R}^{n \times n}$, sparsification removes edges by recovering a binary matrix $P \in \mathbb{B}^{n \times n}$ where $P_{ij} = 1$ indicates that an edge is present between sample $\mathbf{x}_i$ and $\mathbf{x}_j$ and $P_{ij} = 0$ indicates the edge is absent (assume $P_{ii} = 0$ unless otherwise noted). This article will primarily investigate two graph sparsification algorithms: neighborhood approaches including the k-nearest neighbors algorithm and matching approaches such as b-matching (BM). All such methods operate on the matrix A or, equivalently, the distance matrix $D \in \mathbb{R}^{n \times n}$ obtained from A element-wise as $D_{ij} = \sqrt{A_{ii} + A_{jj} - 2A_{ij}}$ since it is possible to convert a similarity function $k(\cdot)$ into a distance function via $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_j) - 2k(\mathbf{x}_i, \mathbf{x}_j)}$.

2.2. 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 graph connecting k closest samples. A recent study showed the dramatic influences these two different neighborhood methods have on clustering techniques (Maier & Luxburg, 2009). The kNN graph remains the more common approach since it is more adaptive to scale and data density while an improper threshold value in the ϵ-neighborhood graph could result in disconnected components or subgraphs in the dataset or even isolated singleton vertices (Figure 1 (b)). In this paper, kNN will be used for neighborhood graph construction since the ϵ-neighborhood graphs provide consistently weaker performance.

The k-nearest neighbor graph is a graph in which two vertices i and j are connected by an edge if the distance D_{ij} between i and j is k-th smallest among the distances distances from i to other objects in $V \setminus i$. Roughly speaking, the k-nearest neighbors algorithm starts with a matrix $\hat{P}$ 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 $\hat{P}_{ij} = 1$. Finally, the matrix is symmetrized as follows $P_{ij} = \max(\hat{P}_{ij}, \hat{P}_{ji})$. It is straightforward to show that k-nearest neighbors is solving the following optimization problem:

$$\min_{\hat{P} \in \mathbb{B}} \sum_{ij} \hat{P}_{ij} D_{ij} \qquad (1)$$
$$s.t. \sum_j \hat{P}_{ij} = k, \hat{P}_{ii} = 0, \forall i, j \in 1, \ldots, n$$

and then produces the solution $P = \max(\hat{P}, \hat{P}^\top)$. 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 adhoc maximization over $\hat{P}$ and its transpose[2], the solution P it produces does not satisfy the equality $\sum_k P_{ij} = k$ but rather only satisfies the inequality $\sum_j P_{ij} \geq k$. Ironically, despite conventional wisdom and the nomenclature, the k-nearest-neighbors algorithm is producing an undirected subgraph with more than k neighbors for each node. It behooves the practitioner to consider the b-matching algorithm which actually achieves the desired output.

2.3. Sparsification via b-Matching

The b-matching problem generalizes maximum weight matching (linear assignment problem), where the objective is to find the binary matrix to minimize the following optimization problem:

$$\min_{P \in \mathbb{B}} \sum_{ij} P_{ij} D_{ij} \qquad (2)$$
$$s.t. \sum_j P_{ij} = b, P_{ii} = 0, P_{ij} = P_{ji}, \forall i, j \in 1, \ldots, n$$

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

achieving symmetry directly without post-processing. Here, the symmetric solution is recovered up-front by enforcing the additional constraints $P_{ij} = P_{ji}$. The matrix then satisfies the equality $\sum_j P_{ij} = \sum_i P_{ij} = b$ strictly. The solution to Eq. 2 is not quite as straightforward or efficient to obtain as the greedy k-nearest-neighbors algorithm. A polynomial time $\mathcal{O}(bn^3)$ solution has been known yet recent advances show much faster variants are possible via (guaranteed) loopy belief propagation (Huang & Jebara, 2007). In Figure 1, an intuitive demonstration of neighborhood graphs and b-matching graph are showed, where appropriate values of ϵ were chosen to make the total number of edges in ϵ-neighborhood graphs comparable with the other graphs. Compared with the neighborhood graph, the b-matching graph is balanced or b-regular. In other words, each vertex in the b-matched graph has only b edges connecting it to other vertices. This advantage plays a key role when conducting label propagation on typical samples $\mathcal{X}$ which are unevenly and non-uniformly distributed. An efficient implementation for the b-matching problem is available online from the authors of the loopy belief propagation method (Huang & Jebara, 2007), which handles both unipartite and bipartite graphs. Previous work also applied b-matching to remove spurious edges and sparsify a graph from the original fully connected adjacency matrix. For instance, in spectral clustering analysis, b-matching showed empirical advantages (Jebara & Shchogolev, 2006). This article focuses on the application of maximum weight b-matching for sparsification and then semi-supervised learning. In addition, due to the faster belief propagation implementation (which is detailed in the Appendix), it is possible to apply the method to large scale problems without significantly increasing the runtime of the overall semi-supervised learning procedure.

Once a graph has been sparsified and a binary matrix P is available to delete unwanted edges, several procedures can then be used to recompute the weights originally in the matrix A to produce a final set of edge weights W.

2.4. Graph Edge Re-Weighting

Given the kernel matrix A and the binary linkage information P from the previous sections, a number of edge weighting schemes are then applied to estimate the weights W of the the now sparse graph. Whenever $P_{ij} = 0$, the edge weight is also $W_{ij} = 0$ however $P_{ij} = 1$ implies that $W_{ij} \geq 0$. This article considers three possible approaches for estimating the non-zero components of W.

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

An alternative approach is *Gaussian kernel* (GK) 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:

$$W_{ij} = P_{ij} \exp\left(-\frac{d(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma^2}\right) \tag{3}$$

where the function $d(\mathbf{x}_i, \mathbf{x}_j)$ evaluates the dissimilarity of sample $\mathbf{x}_i$ and $\mathbf{x}_j$ and σ is the kernel bandwidth parameter. There is are many choices for the distance function $d(\cdot)$ including any ℓ_p distance and the χ^2 distance, as listed below:

$$d_1(\mathbf{x}_i, \mathbf{x}_j) = |\mathbf{x}_i - \mathbf{x}_j| \quad d_2(\mathbf{x}_i, \mathbf{x}_j) = \| \mathbf{x}_i - \mathbf{x}_j \| \tag{4}$$
$$d_3(\mathbf{x}_i, \mathbf{x}_j) = \sum_k \frac{(\mathbf{x}_{i,k} - \mathbf{x}_{j,k})^2}{\mathbf{x}_{i,k} + \mathbf{x}_{j,k}}$$

The ℓ_2 distance is widely used in previous research (Zhu, 2005), while χ^2 distance is useful for histograms (Zhang et al., 2007). Moreover, the cosine distance is another straightforward way to compute scale invariant sample similarly and is commonly used for the task of text classification (Belkin et al., 2005).

One final way of estimating edge weight for the connected edges is motivated by the locally linear embedding technique presented by (Roweis & Saul, 2000). In this procedure, a novel edge weighting model using the non-negative coefficients of the *locally linear reconstruction* (LLR) is performed as explicated in (Wang & Zhang, 2008). Given the sparse connectivity matrix P, the error of a locally linear reconstruction $\mathbf{x}_i$ given its (sparse) connectivity information (or neighborhood information, equivalently) is defined as:

$$\varepsilon_i = \|\mathbf{x}_i - \sum_{j=1}^{n} P_{ij} W_{ij} \mathbf{x}_j\|^2. \tag{5}$$

The best local linear reconstruction can be achieved by minimizing the above reconstruction error. Since direct optimization on Eq. 5 could generate negative coefficients for W_{ij}, constraints are imposed on the optimization such that the weights are non-negative and normalize to unity as follows:

$$\min_{W} \sum_{i} \|\mathbf{x}_i - \sum_{j=1}^{n} P_{ij} W_{ij} \mathbf{x}_j\|^2 \qquad (6)$$
$$s.t. \sum\nolimits_j W_{ij} = 1, W_{ij} \geq 0.$$

By considering all neighborhoods for each point as determined by P_{ij}, it is possible to reconstruct an entire set of edge weights. This gives another procedure for setting the edge weights W from the sparsified binary connectivity P. Also, note that in all three edge weighting procedures, a value of $P_{ij} = 0$ implies that $W_{ij} = 0$.

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

3. Label Diffusion and Inference

Given the constructed graph $\mathcal{G} = \{V, E\}$, whose geometric structure is represented by the weight matrix W, the label inference task is to diffuse the known labels $\mathcal{Y}_l$ to all the unlabeled vertices V_u in the graph and estimate $\hat{\mathcal{Y}}_u$. Designing a robust label diffusion algorithm for such graphs is a widely studied problem and many recent methods are surveyed in (Zhu, 2005; Wang & Zhang, 2008).

In most of these SSL approaches, several standard quantities are computed from the edge weights W, for instance the diagonal node degree $\mathcal{D} = \{d_{ii}\}$, where $d_{ii} = \sum_i W_{ij}$, the graph Laplacian $\triangle = \mathcal{D} - W$, and the normalized graph Laplacian $L = \mathcal{D}^{-1/2} \triangle \mathcal{D}^{-1/2}$. The label information is formulated as a label matrix $Y \in \mathbb{B}^{|V| \times c}$, where $Y_{ij} = 1$ if sample $\mathbf{x}_i$ is associated with label j for $j \in \{1, 2, \cdots c\}$ and $Y_{ij} = 0$ otherwise. For single label problems (as opposed to multi-label problems), the constraints $\sum_j Y_{ij} = 1$ are also imposed. The SSL methods then utilize the graph and W as well as the known labels to recover a continuous classification function $F = [F_l \;\; F_u]^\top \in \mathbb{R}^{|V| \times c}$ by optimizing a predefined energy cost on the graph. A number of approaches for recovering F will be considered and are summarized below.

Gaussian Random Fields (*GRF*): (Zhu et al., 2003) proposed Gaussian random fields and harmonic functions for optimizing the following cost on a weighted graph $\mathcal{G}$ to recover the classification function:

$$\min_{F \in \mathbb{R}^{|V| \times c}} \operatorname{tr}(F^\top \triangle F). \qquad (7)$$

Two conditions are imposed on the harmonic function F, $\triangle F_u = 0$ for unlabeled samples, and $F_l = Y_l$ on labeled data.

Local and Global Consistency (*LGC*): Instead of clamping the classification function on labeled nodes by setting the hard constraint $F_l = Y_l$, (Zhou et al., 2004) presented an elastic fitness term to regularize the energy function as follows:

$$\min_{F \in \mathbb{R}^{|V| \times c}} \operatorname{tr}\{F^\top L F + \mu (F - Y)^\top (F - Y)\}. \qquad (8)$$

where the parameter $\mu \in [0 \;\; \infty)$ balances the tradeoff between local fitting and global smoothness of the function F.

Graph Transduction via Alternating Minimization (*GTAM*): Both the above label diffusion methods depend on a univariate energy function and treat the classification function F as the only variable of interest. Since the given labels launch the label diffusion process, the above methods are extremely sensitive to the choice (and noisiness) of the initially provided labels. To alleviate the dependency on the initial labels, (Wang et al., 2008) proposed a bivariate transductive formulation which alternates optimization of both the classification function F and the predicted binary label variables $\mathcal{Y}_u$ or Y_u in matrix form. Specifically, the energy function is defined as:

$$\min_{\substack{F \in \mathbb{R}^{|V| \times c} \\ Y \in \mathbb{B}^{|V| \times c}}} \operatorname{tr}\left\{F^\top L F + \mu (F - VF)^\top (F - VY)\right\}$$
$$s.t. \quad \sum\nolimits_j Y_{ij} = 1 \qquad (9)$$

where V is the label regularization matrix, which modulates label importance and class ratios. This transductive graph labeling approach provided significant improvements in accuracy (Wang et al., 2008).

These three label prediction methods will be combined with the variety of graph sparsification and edge reweighting procedures in Table 1 to determine in a thorough manner which configurations and algorithms are best suited for semi-supervised learning in practice. The table shows the shorthand notation that will be used to refer to the various combinations of sparsification and edge-reweighting schemes that will be considered.

Table 1. The graph construction using different types of sparsification and re-weighting methods.

	Binary	Gaussian Kernel	Locally Linear Reconstruction
kNN	KNN-BN	KNN-GK	KNN-LLR
b-Matching	BM-BN	BM-GK	BM-LLR

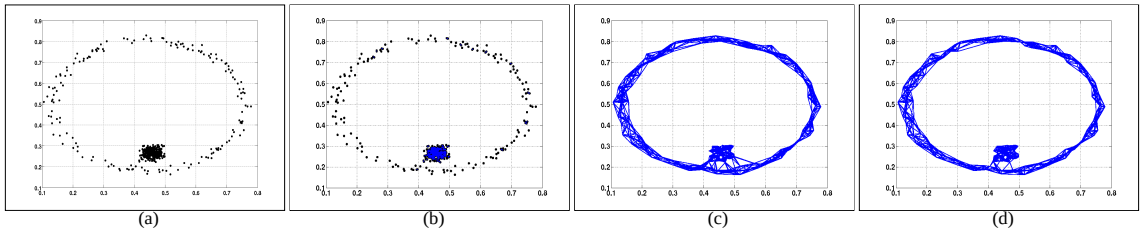


Figure 1. The synthetic dataset used for evaluating the graph construction methods. a) The synthetic data; b) the ϵ-neighborhood graph; c) the kNN graph ($k = 10$); d) the b-matched graph ($b = 10$).

4. Experiments

4.1. Synthetic Data

The synthetic dataset used in these experiments is shown in Figure 1 (a). Clearly, this dataset 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.

The three label prediction approaches, *GRF*, *LGC* and *GTAM*, were investigated under different graph sparsification and edge reweighting methods. Note that the ℓ_2 distance was used in the Gaussian kernel weighting (see Table 1). The experiment consisted of 50 folds or runs where each fold involved labeling the rest of the dataset given a single positive and a single negative example drawn from each class randomly. The average error rate over the 50 folds is reported in Figure 2 for different graphs construction and labeling algorithms. The figure shows that the b-matching graph significantly improved stability and robustness across a wide range of choices for b. It improves over the kNN graph under all the weighting models and all label diffusion approaches. When the appropriate values of k and b are selected, the simple binary b-matching graph provides better performance than the kNN graph with more sophisticated weighting approaches. A demonstration of the constructed graphs by b-matching and neighborhood connection models (kNN and ϵ-neighborhood) is shown in Figure 1 (b)(c)(d), where more cross-cluster edges are created by the kNN approach due to the unevenly distributed sample points. Note that ϵ-neighborhood graphs were not included in some experiments because of overall weak performance. This is because they have a tendency to fragment the dataset into disconnected components which makes the performance of SSL deteriorate. In fact, most edges in the ϵ-neighborhood graphs appeared only in the central dense cluster in the synthetic dataset across most choices of ϵ, as shown in Figure 1 (b).

In another experiment, we evaluate the robustness of the b-matched graph and the kNN graph with respect to the Gaussian kernel size σ. Since *GTAM* was fairly robust to the choice of σ, only the experiments using the *LGC* and *GRF* methods were reported here. The prediction errors for different values of σ are shown in Figure 3. Clearly, the b-matched graph is much more robust to the kernel bandwidth parameter for both the *LGC* and *GRF* methods.

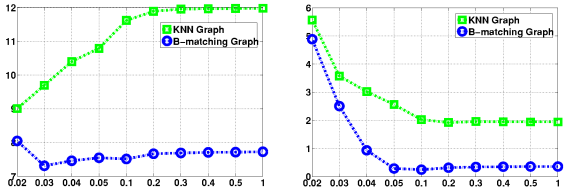
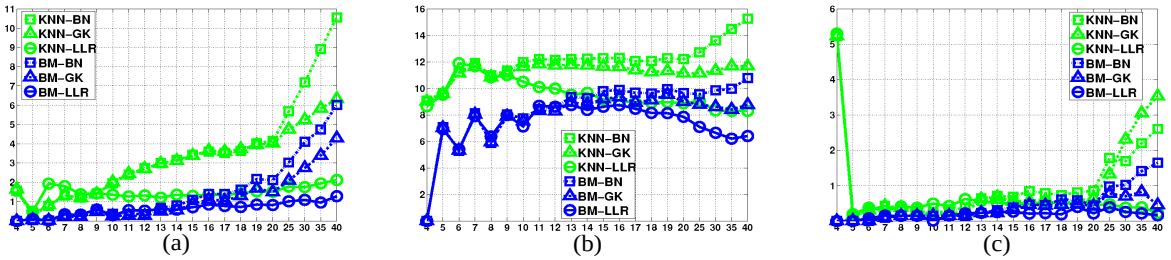


Figure 3. Robustness to different kernel bandwidth settings using the Gaussian kernel weighting. The x axis shows the kernel bandwidth parameter and the y axis shows the error rate (%) for a) the *LGC* method, and b) the *GRF* method.

4.2. Real Benchmark Data

To evaluate the performance of SSL approaches across the wide range of graph construction methods, we carried out experiments on real benchmark data sets (Chapelle et al., 2006). Two datasets (USPS and TEXT) were used in these experiments. Moreover, we used the originally suggested data splits to conduct our experiments for fair comparison, where each data set is associated with 12 different partitions of labeled and unlabeled subsets. For all experiments, both 10 and 100 labeled samples are tested. The parameters for these experiments are uniformly set as $k = b = 12$, $\mu = 0.05$. The kernel size for GK reweighting is $\sigma = \bar{d}_k/3$, where $\bar{d}_k$ is the average distance between each sample and its kth nearest neighbor (Chapelle et al., 2006). In addition, the ℓ_2 distance was used for the USPS data and the χ^2 distance was used for Text data. The experimental results are shown (in terms of average error rate) in Table 2.

In the table, 5 more methods were also evaluated in addition to the techniques mentioned in this article since they were among the best performers (out of a total of 13 methods) reported in the recent sur-



(a) (b) (c)

Figure 2. Robustness to different values of k in the kNN graph and b in the b-matched graph. The x axis explores various settings for k or b while the y axis reports the error rate (%) of a) the *GRF* method, b) the *LGC* method, and c) the *GTAM* method.

vey paper (Chapelle et al., 2006). Along with these 5 competitor techniques, the three label diffusion algorithms *GRF*, *LGC*, and *GTAM* were tested using different graphs construction methods. From the table of results, some statistically significant conclusions can be made. First, the b-matched graph consistently improves performance compared with the kNN graph in most of the cases. A drastic example is seen in the improvement b-matching brings to *GTAM* on the TEXT dataset. Second, the *GTAM* method had consistently better accuracy than the other label diffusion methods (*LGC* and *GRF*) which otherwise had comparable accuracy (in other words $GTAM \gg LGC \simeq GRF$).

Table 2. Experimental results on the benchmark data sets (in terms of % error rate) for the variety of label prediction algorithms as well as graph construction approaches.

Data set	USPS		TEXT	
# of labels	10	100	10	100
QC + CMN	13.61	6.36	40.79	25.71
TSVM	25.2	9.77	31.21	24.52
LDS	17.57	4.96	27.15	23.15
Laplacian RLS	18.99	4.68	33.68	23.57
CHM (normed)	20.53	7.65	-	-
GRF-KNN-BN	19.11	9.07	47.65	41.56
GRF-KNN-GK	13.01	5.58	48.2	41.57
GRF-KNN-LLR	19.20	11.17	47.14	35.17
GRF-BM-BN	18.98	9.06	43.16	25.27
GRF-BM-GK	12.93	5.34	42.96	25.29
GRF-BM-LLR	18.96	10.08	42.95	24.56
LGC-KNN-BN	14.99	12.34	48.63	43.44
LGC-KNN-GK	12.34	5.49	49.06	41.51
LGC-KNN-LLR	15.88	13.63	44.88	37.52
LGC-BM-BN	14.62	11.71	40.88	26.19
LGC-BM-GK	11.92	5.21	41.32	23.85
LGC-BM-LLR	14.67	12.19	40.27	24.92
GTAM-KNN-BN	6.59	5.98	49.36	46.67
GTAM-KNN-GK	4.86	**2.56**	49.07	46.06
GTAM-KNN-LLR	6.77	6.19	41.42	39.59
GTAM-BM-BN	6.00	5.08	17.74	16.78
GTAM-BM-GK	**4.62**	3.08	19.73	17.89
GTAM-BM-LLR	5.59	5.14	**16.06**	**14.87**

5. Conclusion

Though a gamut of graph-based semi-supervised learning algorithms has been developed in the past years, the influence of graph construction procedures on such algorithms has only received limited study despite its critical impact on accuracy (Wang & Zhang, 2008). Most implementations of SSL apply neighborhood methods (such as k nearest neighbors) to create graphs for semi-supervised learning, yet this article shows that these can produce imbalanced and irregular graphs both in synthetic and real data situations. This article proposed the maximum weight b-matching method for graph construction (or sparsification) and showed its consistent empirical advantage across a wide range of algorithms, weighting procedures, and datasets. In addition, the article systematically studied the impact of graph construction approaches on a variety of semi-supervised algorithms. This study confirms the importance of graph construction methods in semi-supervised learning as well as motivates b-matching as a valuable alternative to k-nearest neighbors which many practitioners believe produces regular undirected graphs yet in practice often generates irregular graphs. While theoretical guarantees for the advantages of b-matching are not provided in this article they are the subject of future work.

A. b-Matching via Belief Propagation

Given an adjacency matrix $C \in \mathbb{R}^{n \times n}$ of a graph $\mathcal{G} = (V, E)$ with n nodes V and $\mathcal{O}(n^2)$ edges E, the maximum weight b-matching problem finds a subgraph of $\mathcal{G}$ with maximum weight while constraining the number of edges for each vertex to equal b. b-matching is a direct generalization of the maximum weight unipartite matching problem ($b = 1$) which is solved by Edmonds' algorithm in polynomial time $\mathcal{O}(n^3)$ (Edmonds, 1965). In (Bayati et al., 2005), the 1-matching problem was formulated as a discrete probability distribution and solved by max-product loopy belief propagation (BP) to recover the maximum a posteriori (MAP) assignment in $\mathcal{O}(n^3)$. In (Huang & Jebara, 2007), the extension of loopy belief propagation from

1-matching to b-matching was provided as well as a proof of convergence in $\mathcal{O}(bn^3)$. Furthermore, various implementation issues and changes to the canonical message passing rules of belief propagation were provided to maintain computational efficiency (standard message passing involves messages of exponential size) leading to an efficient tool to solve the combinatorial b-matching problems. Here we show the algorithm in the bipartite case where the nodes in the graph are split into two sets a priori before being b-matched. The extension to the unipartite case is straightforward simply by modifying the algorithm such that messages are passed between all pairs of nodes instead of only in across the bipartition.

Assume that b-matching returns the neighbor vertex sets $\mathcal{N}(u_i)$ and $\mathcal{N}(v_j)$ for vertex u_i and v_j, respectively. The combinatorial problem in Eq. 2 can be written as:

$$\max_{\mathcal{N}} \mathcal{W}(\mathcal{N}) = \max_{\mathcal{N}} \sum_{i=1}^{|V|} \sum_{v_k \in \mathcal{N}(u_i)} C_{ik} + \sum_{j=1}^{|V|} \sum_{u_l \in \mathcal{N}(v_j)} C_{lj} \quad (10)$$

For each vertex, two random variables are defined as $z_i \in Z$ and $s_j \in S$ and $z_i = \mathcal{N}(u_i)$ and $s_j = \mathcal{N}(v_j)$. Hence we can have the following potential functions:

$$\phi(z_i) = \exp(\sum_{v_j \in z_i} C_{ij}) \quad \phi(s_j) = \exp(\sum_{u_i \in s_j} C_{ij})$$
$$\psi(z_i, s_j) = \neg(v_j \in z_i \oplus u_i \in s_j). \quad (11)$$

By multiplying the above potential functions and pairwise clique functions, the objective function for weighted b-matching problem can be formulated as a probability distribution via $p(Z, S) \propto \exp(\mathcal{W}(\mathcal{N}))$ (Chung, 1997), where the joint distribution can be expressed as:

$$p(Z, S) = \frac{1}{Z} \prod_{i=1}^{|V|} \prod_{j=1}^{|V|} \psi(z_i, s_j) \prod_{k=1}^{|V|} \phi(z_i)\phi(s_j) \quad (12)$$

Max-product message passing on the above distribution is guaranteed to converge to the true maximum in $\mathcal{O}(n^3)$ time on bipartite graphs, the proof provided in (Huang & Jebara, 2007) is omitted here for brevity. In practice, convergence is much faster than this worst case runtime. Furthermore, (Salez & Shah, 2009) prove that, under mild assumptions on the matrix C, belief propagation for matching problems can converge in $\mathcal{O}(n^2)$ for dense graphs. If the graph is sparse, convergence is typically $\mathcal{O}(|E|)$ time or proportional to the number of edges in the graph.

References

Bayati, M., Shah, D., & Sharma, M. (2005). Maximum weight matching via max-product belief propagation. *Int. Symp. on Information Theory* (pp. 1763–1767).

Belkin, M., & Niyogi, P. (2008). Towards a Theoretical Foundation for Laplacian Based Manifold Methods. *J. Comput. System Sci.*, 1289–1308.

Belkin, M., Niyogi, P., & Sindhwani, V. (2005). On manifold regularization. *Int. Workshop on Artificial Intelligence and Statistics*.

Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. *Int. Conf. on Mach. Learn.* (pp. 19–26).

Chapelle, O., Schölkopf, B., & Zien, A. (Eds.). (2006). *Semi-supervised learning*. Cambridge, MA: MIT Press.

Chung, F. (1997). *Spectral Graph Theory*. American Mathematical Society.

Edmonds, J. (1965). Paths, trees and flowers. *Canadian Journal of Mathematics*, *17*, 449–467.

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

Jebara, T., & Shchogolev, V. (2006). B-Matching for Spectral Clustering. *The European Conf. on Mach. Learn.* (pp. 679–686). Springer.

Maier, M., & Luxburg, U. (2009). Influence of graph construction on graph-based clustering measures. *The Neural Information Processing Systems*, *22*, 1025–1032.

Roweis, S., & Saul, L. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, *290*, 2323–2326.

Salez, J., & Shah, D. (2009). Optimality of Belief Propagation for Random Assignment Problem. *ACM-SIAM Symp. on Discrete Algorithms*.

Wang, F., & Zhang, C. S. (2008). Label propagation through linear neighborhoods. *IEEE Trans. Knowl. Data Eng.*, *20*, 55–67.

Wang, J., Jebara, T., & Chang, S. F. (2008). Graph transduction via alternating minimization. *Int. Conf. on Mach. Learn.* (pp. 1144–1151).

Zhang, J., Marszalek, M., Lazebnik, S., & Schmid, C. (2007). Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *Int. J. Compt. Vision.*, *73*, 213–238.

Zhou, D., Bousquet, O., Lal, T., Weston, J., & Schölkopf, B. (2004). Learning with local and global consistency. *The Neural Information Processing Systems* (pp. 321–328).

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

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *Int. Conf. on Mach. Learn.* (pp. 912–919).