8-27-2009

Topics in Graph Construction for Semi-Supervised Learning

Partha Pratim Talukdar
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-09-13.

Topics in Graph Construction for Semi-Supervised Learning

Abstract

Graph-based Semi-Supervised Learning (SSL) methods have had empirical success in a variety of domains, ranging from natural language processing to bioinformatics. Such methods consist of two phases. In the first phase, a graph is constructed from the available data; in the second phase labels are inferred for unlabeled nodes in the constructed graph. While many algorithms have been developed for label inference, thus far little attention has been paid to the crucial graph construction phase and only recently has the importance of the graph construction for the resulting success in label inference been recognized. In this report, we shall review some of the recently proposed graph construction methods for graph-based SSL. We shall also present suggestions for future research in this area.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-09-13.

Topics in Graph Construction for Semi-Supervised Learning

Partha Pratim Talukdar
University of Pennsylvania
partha@cis.upenn.edu

August 27, 2009

Contents

Acknowledgment

I would like to thank my committee Mark Liberman, Ben Taskar (chair), and Lyle Ungar. Special thanks to Ted Sandler for many useful discussions.

Abstract

Graph-based Semi-Supervised Learning (SSL) methods have had empirical success in a variety of domains, ranging from natural language processing to bioinformatics. Such methods consist of two phases. In the first phase, a graph is constructed from the available data; in the second phase labels are inferred for unlabeled nodes in the constructed graph. While many algorithms have been developed for label inference, thus far little attention has been paid to the crucial graph construction phase and only recently has the importance of the graph construction for the resulting success in label inference been recognized. In this report, we shall review some of the recently proposed graph construction methods for graph-based SSL. We shall also present suggestions for future research in this area.

1 Introduction

Semi-Supervised Learning (SSL) methods can learn from labeled data combined with unlabeled data. While labeled data is hard and expensive to obtain, unlabeled data is often widely available. Because of this desirable property, SSL methods have received considerable attention in recent years (Chapelle et al., 2006). Graph-based SSL methods are particularly appealing because of their empirical success along with computational efficiency (Zhu et al., 2003; Zhou et al., 2004; Belkin et al., 2005; Bengio et al., 2007; Wang et al., 2008; Subramanya and Bilmes, 2008; Jebara et al., 2009; Talukdar and Crammer, 2009). Most of these methods are transductive in nature i.e. they can't be used to classify an unseen test point in the future, though exceptions exist (Belkin et al., 2005). Graph-based SSL methods have been been effective in a wide variety of domains ranging from video recommendation (Baluja et al., 2008), protein classification (Tsuda et al., 2005) to assignment of semantic types to entities (Talukdar et al., 2008). Theoretical justification for graph-based methods is an area of active research (Lafferty and Wasserman, 2007; Niyogi, 2008; Singh et al., 2009).

Graph-based SSL methods operate on a graph where a node corresponds to a data instance and a pair of nodes are connected by a weighted edge. A few nodes in the graph are assigned labels. This initial supervision separates graph-based SSL methods from spectral clustering methods (von Luxburg, 2007) which are unsupervised. Staring with this supervision, a graph-based SSL algorithm assigns labels to other unlabeled nodes in the graph. In most cases, the objective for label inference optimized by the algorithm results in an iterative process which amounts to spreading labels from the labeled to unlabeled nodes over the graph structure. Hence, graph-based SSL methods are sometimes also called *Label Propagation* methods. An alternative interpretation is to view these methods as performing random walks on the graph (Zhu et al., 2003; Baluja et al., 2008; Talukdar and Crammer, 2009).

As is evident from the discussion above, the graph on which learning is performed is a central object for any graph-based SSL method. In many real world domains (e.g. the web, social networks, citation networks, etc.), the data is relational in nature and there is already an implicit underlying graph. Graph-based methods are a natural fit in these domains. However, for a majority of learning tasks, the data instances are assumed to be independent and identically distributed (i.i.d.). In such cases, there is no explicit graph structure to start with. The common practice is to create a graph in the first step from independent data instances, and in the second step apply one of the graph-based SSL methods on the constructed graph. Even though graph-based SSL methods have been successfully applied to these tasks (Zhu et al., 2003; Subramanya and Bilmes, 2008), very

little attention has been given to the graph construction part, with majority of the research focus devoted to the post-graph construction learning algorithms. Only recently, importance of graph construction for resulting success in label inference has begun to be recognized (Maier et al., 2009; Wang and Zhang, 2008; Daitch et al., 2009; Jebara et al., 2009). In this survey, we shall review some of these methods for graph construction and also suggest opportunities for future research.

In Section 2, we shall present the notations and terminologies used throughout the report. Also, we shall present the assumptions common to all the methods reviewed.

In Section 3, we shall review two commonly used graph-construction methods: k-Nearest Neighbor (k-NN) and ϵ-neighborhood. We shall briefly look at their properties and also list advantages of k-NN over ϵ-neighborhood.

In Section 4, we shall review the recently proposed method of graph construction using b-matching (Jebara et al., 2009). The idea will be to induce regular graphs where all nodes have the same degree (b). In contrast, popular methods like k-NN can generate irregular graphs. Through a variety of experiments, Jebara et al. (2009) argue for the importance of regular graphs for graph-based SSL methods. The graph construction process in (Jebara et al., 2009) involves two stages: in the first step a sparse graph is constructed where all edges have weight 1. In the second step, the edges are re-weighted using the Locally Linear Reconstruction (Roweis and Saul, 2000), among others.

In Section 5, we shall review another recently proposed method for graph construction (Daitch et al., 2009). In (Daitch et al., 2009), the concepts of *Hard* and α-*Soft* graphs are introduced. In case of hard graphs, each node is required to have a minimum weighted degree of 1. While, in case of α-soft graphs, this degree requirement is relaxed to allow some nodes (e.g. outliers) have lesser degree. Daitch et al. (2009) also analyze properties of the induced graphs, which is probably one of the first analyses of its kind. Effectiveness of these graphs in a variety of experiments are also presented in (Daitch et al., 2009) which we shall briefly review. We shall also explore the relationship between the method in (Daitch et al., 2009) to that of (Wang and Zhang, 2008).

In Section 6, we shall look at the method of kernel-alignment based spectral kernel design (Zhu et al., 2005). The idea here is to construct a kernel to be used for supervised learning, starting with a fixed graph and in the process transforming the spectrum of the graph Laplacian of the fixed graph. Though this is not exactly a graph construction method, it provides interesting insights into the graph Laplacian and its importance for graph-based SSL, which in turn may be exploited for graph construction. We shall also briefly look at the relationship between (Zhu et al., 2005) and (Johnson and Zhang, 2008). In (Johnson and Zhang, 2008), the relationship between graph-based SSL and standard kernel-based supervised learning is explored.

Finally, in Section 7, we shall analyze the methods further and also suggest promising directions for future research.

2 Problem Statement

We are given a set of n points, $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, sampled from a d dimensional space. We shall interchangeably use the terms *points*, *instances* and *vectors*. Each point has a length-c label vector associated with it, where c is the total number of classes. The complete label set is given by $Y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$. Labels for the first l points, $X_l = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l\}$, are known. Ultimately,

we are interested in inferring labels for the remaining $n - l$ points, $X_u = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_n\}$. As mentioned in Section 1, graph-based methods operate on a graph where each point is associated with a node. In this report, we shall concentrate on the problem of constructing a graph on which the above mentioned methods could be applied to learn labels for the points in X_u.

We shall represent the graph by $G = (V, E, W)$, where V is the set of nodes ($|V| = n$), E is the set of edges ($|E| = m$) and W is the $n \times n$ edge weight matrix where weight of the edge (i, j) is given by W_{ij}. In the constructed graph G, the node corresponding to point $\mathbf{x}_i$ will be called $\mathbf{x}_i$. In such a graph, a weighted edge will connect two neighboring nodes. Unless otherwise stated, we shall assume the following:

- The induced graph is undirected and that all edge weights are non-negative i.e. $W_{ij} = W_{ji}$ and $W_{ij} \geq 0,\ \forall i, j$
- $W_{ij} = 0$ indicates the absence of an edge between nodes i and j.
- There are no self-loops i.e. $W_{ii} = 0,\ \forall i = 1, \ldots, n$.

Given that the nodes (V) are fixed, the real problem is to learn the edge weight matrix W subject to the constraints above. Some of the methods surveyed may impose additional restrictions on W which we shall specify as we go along.

3 k-Nearest Neighbor and ϵ-Neighborhood Methods

k-Nearest Neighbors (k-NN) and ϵ-neighborhood are some of the most popular methods for constructing a sparse subgraph from a set of points. Both methods assume availability of a similarity or kernel function using which similarity (or distance) between any two points in the data can be computed. In case of k-NN, undirected edges between a node and its k-nearest neighbors are greedily added. Finding the k-nearest neighbors in a naive way can be computationally expensive, specially in case of large datasets. However, a variety of alternatives and approximations have been developed over the years (Bentley, 1980; Beygelzimer et al., 2006).

In ϵ-neighborhood based graph construction, an undirected edge between two nodes is added only of the distance between them is smaller than ϵ, where $\epsilon > 0$ is a predefined constant. In other words, given a point $\mathbf{x}$, a ball (specific to the chosen distance metric) of radius ϵ is drawn around it and an undirected edge between $\mathbf{x}$ and all points inside this ball are added.

k-NN methods enjoy certain favorable properties compared to ϵ-neighborhood based graphs. For example, k-NN methods are adaptive to scale and density while an inaccurate choice of ϵ may result in disconnected graphs (Jebara et al., 2009). An empirical verification of this phenomenon on a synthetic dataset is shown in Figure 1. Also, k-NN methods tend to perform better in practice than ϵ-neighborhood based methods (Jebara et al., 2009). Recently, Maier et al. (2009) show that application of the same clustering algorithm (Normalized Cut, in this case) on two different graphs constructed using k-NN and ϵ-neighborhood may lead to systematically different clustering criteria.

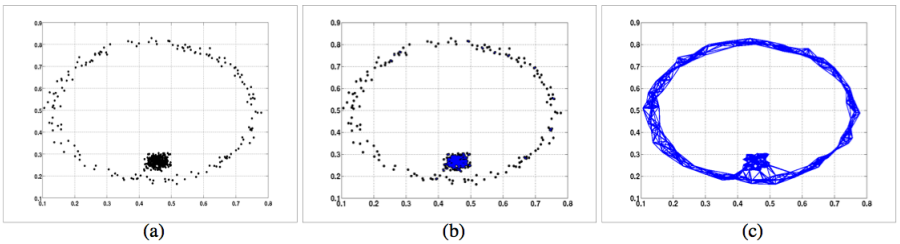

Figure 1: k-NN and ϵ-neighborhood graphs constructed from a synthetic dataset (Jebara et al., 2009) (a) The synthetic dataset, (b) ϵ-neighborhood graph, (c) k-NN graph (k = 10). The ϵ-neighborhood is quite sensitive to the choice of ϵ and it may return graphs with disconnected components as in (b).

4 Graph Construction using b-Matching (Jebara et al., 2009)

As we had discussed previously in Section 3, k-NN methods are popular in constructing graphs. However, contrary to its name, k-NN methods often generate graphs where different nodes have different degrees. A node v can be in the k-nearest neighborhood of more than k nodes, which in itself results in v having degree greater than k. From example, consider Figure 2 which shows the 1-NN graph of a set of 5 points arranged in specific configuration. We note that the central node ends up with degree 4 which is significantly higher than degree 1 of all other four nodes. In order

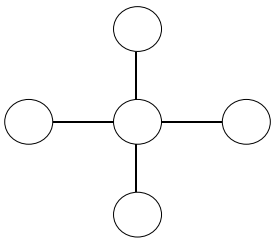

Figure 2: 1-NN graph for a set of 5 points. Note that the central node ends up with degree 4 while all other nodes have degree 1, resulting in an irregular graph.

to induce a regular graph, a b-matching based graph construction process is proposed in (Jebara et al., 2009). b-matching guarantees a regular graph in which each vertex has exactly b degree.

We shall use the same notations as in Section 2. The graph construction process outlined in (Jebara et al., 2009) can be decomposed into the following two steps:

- Graph Sparsification: Given a set of n points with similarity defined between each pair of points, a total of $\binom{n}{2}$ edges (undirected) are possible. This will result in a very dense graph

which can significantly slow down any subsequent inference on this graph. Hence, during graph sparsification a subset of the edges are selected which will be present in the final graph.

- Edge Re-Weighting: During this step, weights are learned for the edges selected in the sparsification step.

4.1 Graph Sparsification

Let $k(\cdot,\cdot)$ be a kernel function used to measure similarity between two points. We define the similarity matrix $A \in \mathbb{R}^{n\times n}$ where $A_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Note that A is a dense and symmetric (as k is symmetric). Sparsification removes edges by estimating a matrix $P \in B$ (B is the space of $\{0,1\}^{n\times n}$ matrices) where $P_{ij} = 1$ implies an edge between points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the final graph and $P_{ij} = 0$ implies absence of an edge between these two points in the final graph. We set $P_{ii} = 0$, i.e. there are no self-loops. A symmetric distance matrix $D \in \mathbb{R}^{n\times n}$ can be constructed out of A using the following transformation

$$D_{ij} = \sqrt{A_{ii} + A_{jj} - 2A_{ij}}$$

(Jebara et al., 2009) note that k-nearest neighbor can be viewed as a graph sparsification process which is optimizing the following optimization problem

$$\begin{gathered}\min_{\hat{P}\in B} \textstyle\sum_{ij} \hat{P}_{ij} D_{ij} \\ \text{s.t. } \textstyle\sum_j \hat{P}_{ij} = k,\ \hat{P}_{ii} = 0,\ \forall i,j \in 1,\ldots,n\end{gathered}$$

where the final binary matrix P is obtained by the symmetrization step: $P_{ij} = \max(\hat{P}_{ij}, \hat{P}_{ji})$. Even though the optimization satisfies the $\sum_j \hat{P}_{ij} = k$ constraint, the subsequent symmetrization step only enforces the constraint $\sum_j P_{ij} \geq k$, as we had discussed at the beginning of this section. Sparsification by b-matching overcomes this problem by optimizing the following optimization problem

$$\begin{gathered}\min_{P\in B} \textstyle\sum_{ij} P_{ij} D_{ij} \\ \text{s.t. } \textstyle\sum_j P_{ij} = b,\ P_{ii} = 0,\ P_{ij} = P_{ji},\ \forall i,j \in 1,\ldots,n\end{gathered}$$

Please note that the degree ($\sum_j P_{ij} = b$) and symmetrization ($P_{ij} = P_{ji}$) constraints are included directly into the optimization and hence there is no need of any ad-hoc post-processing symmetrization step as in the case of k-NN. Even though the change from k-NN to b-matching may seem quite trivial, there are significant differences in computation complexity. We shall discuss more about it in Section 4.4.

4.2 Edge Re-Weighting

With the set of edges selected through matrix P in the previous section, we would like to estimate the weights of the selected edges. In particular, we would like to estimate the symmetric edge weight matrix W with the following constraint: $W_{ij} \geq 0$ if $P_{ij} = 1$ and $W_{ij} = 0$ if $P_{ij} = 0$. Three edge re-weighting strategies are presented in (Jebara et al., 2009). They are,

- **Binary (BN)**: In this case, we set $W = P$ i.e. weights of all selected edges ($P_{ij} = 1$) are uniformly set to 1. This strategy may prevent recovery from errors committed during estimation of P.

- **Gaussian Kernel (GK)**: In this case, we set

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

  where $d(\mathbf{x}_i, \mathbf{x}_j)$ measures the distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$, and σ is a hyper-parameter. l_p distance (Zhu, 2005), χ^2 distance and cosine distance (Belkin et al., 2005) are some of the choices for $d(\cdot, \cdot)$.

- **Locally Linear Reconstruction (LLR)**: This re-weighting scheme is based on the Locally Linear Embedding (LLE) technique (Roweis and Saul, 2000). Given the binary connectivity matrix P, weights of edges incident on node $\mathbf{x}_i$ (the W_{ij}'s) are estimated by minimizing the following error

$$\epsilon_i = ||\mathbf{x}_i - \sum_j P_{ij} W_{ij} \mathbf{x}_j||^2$$

  In this case, we would like reconstruct $\mathbf{x}_i$ from its neighborhood information with $\sum_j P_{ij} W_{ij} \mathbf{x}_j$ representing the reconstructed vector. The resulting optimization problem is shown below

$$\begin{aligned} &\min_W \textstyle\sum_i ||\mathbf{x}_i - \sum_j P_{ij} W_{ij} \mathbf{x}_j||^2 \\ &\text{s.t. } \textstyle\sum_j W_{ij} = 1,\ W_{ij} \geq 0,\ i = 1, \ldots, n \end{aligned}$$

  In (Jebara et al., 2009), it is claimed that the matrix W obtained from the above optimization is symmetric. However, with the symmetric constraint ($W_{ij} = W_{ji}$) *not* enforced during optimization, it is difficult to see how that may be the case.

Anyways, With the edge weights (W) determined, we now have a weighted graph constructed on which any of the standard graph-based semi-supervised techniques e.g. Gaussian Random Fields (GRF) (Zhu et al., 2003), Local and Global Consistency (LGC) (Zhou et al., 2004) and Graph Transduction via Alternating Minimization (GTAM) (Wang et al., 2008) may be applied.

4.3 Results

Empirical evidence of importance of graph construction and its resulting effect on semi-supervised learning is presented in (Jebara et al., 2009). In particular, graph construction using following six combinations of sparsification and re-weighting methods are compared: { KNN, BM } × { BN, GK, LLR }. For each constructed graph, three transductive learning methods: GRF, LGC and GTAM are compared against their label prediction accuracy on unlabeled nodes. In these experiments, all hyper-parameters (e.g. k, b, σ etc.) are heuristically set. Experimental results on synthetic and two real-world datasets are presented. From these results we observe that in most cases, b-matched graph improved performance compared to k-NN graph. Also, GTAM seems to be effective compared to GRF and LGC, with GRF and LGC achieving comparable accuracy.

4.4 b-matching Complexity

While the greedy algorithm for k-NN is well known, complexity of the known polynomial time algorithm for maximum weight b-matching is $O(bn^3)$. Though recent advances in terms of loopy belief propagation has resulted in faster variants of it (Huang and Jebara, 2007). A sketch of the algorithm for bipartite graphs is shown in Appendix A of (Jebara et al., 2009).

4.5 Discussion

In (Jebara et al., 2009), the sparse matrix P is selected with one similarity measure while the edge re-weights are estimated with respect to another similarity measure (e.g. GK and LLR) resulting in a mismatch which may lead to sub-optimal behavior. It is worthwhile to consider learning both jointly using one similarity measure, and with some added penalty in favor of sparseness.

In (Jebara et al., 2009), the hyper-parameters are heuristically set which may favor one method or the other. In order to have a thorough comparison, a grid search over these parameters results based on those would have been more insightful. Also, Computational complexity is one of the major concerns for b-matching. It would have been insightful to have runtime comparison of b-matching with k-NN.

As suggested in (Jebara et al., 2009), it will be interesting to see theoretical justification for the advantages of b-matching, which is currently a topic of future research.

5 Graph Construction using Local Reconstruction (Daitch et al., 2009)

Given a set of vectors (or points in high dimensional space) $\mathbf{x}_1, \ldots, \mathbf{x}_n$, (Daitch et al., 2009) ask the question: what is the *best* graph that fits these points, where *best* is measured as per some fitness criteria? As an answer to this question, two types of graphs (Hard and α-Soft graphs), their properties and empirical comparison of these graphs to previously proposed graph construction methods are presented in (Daitch et al., 2009). We shall look into details of each of these in this section. We shall change the notations in (Daitch et al., 2009) as needed to make it consistent with previous sections of this report.

5.1 Hard Graphs

As per the definition in (Daitch et al., 2009), *hard graphs* are graphs where each node is strictly required to have (weighted) degree of at least 1. As before, W is the symmetric edge weight matrix where $W_{ij} = W_{ji} \geq 0$. Weighted degree, d_i, of a node i is defined as $d_i = \sum_j W_{ij}$. Self loops are not allowed and hence $W_{ii} = 0, \ \forall i$. The *hard graph* construction algorithm presented in (Daitch et al., 2009) minimizes the optimization problem shown in (1).

$$\min_W f(W) = \sum_i ||d_i \mathbf{x}_i - \sum_j W_{ij} \mathbf{x}_j||^2, \text{ s.t. } d_i \geq 1, i = 1, \ldots, n \tag{1}$$

The degree constraints are necessary as an unconstrained minimization of $f(W)$ will result in $W_{ij} = 0,\ \forall i, j$. We note that $f(W)$ is similar to the LLE (Roweis and Saul, 2000) objective when $d_i = 1\ \forall i$. LLE pre-selcts the neighbors and allows non-symmetric edge weight matrix, where some of the edge weights can be negative. In contrast, the graph construction method presented in Daitch et al. (2009) performs neighborhood selection and edge weighting in a single step, the edge weight matrix is symmetric and only non-negative edge weights are allowed. We shall now look at an algorithm to solve (1).

Let $\mathbf{w}$ be a vector representation of the weight matrix W[1], with only positive weighted edges considered. Note that while W is of size $n \times n$, $\mathbf{w}$ is a vector of length m as we consider only the edges currently present in the graph i.e. edges with positive weight. Let U be a $n \times m$ matrix with $U_{ie} = 1$ and $U_{je} = -1$ for each $e = (i, j) \in E$. Let V be a $m \times m$ diagonal matrix with

$$\text{diag}(V) = \mathbf{w}$$

Matrix V's diagonal stores weights of all the edges in E. Let $x^{(k)}$ be the kth column of the $n \times d$ data matrix X whose ith row represents $\mathbf{x}_i$. We define the vector $z^{(k)} = U^T x^{(k)}$ and its corresponding matrix $Z = \text{diag}(z^{(1)}, \ldots, z^{(d)})$. The Laplacian of the constructed graph, L, may be decomposed as $L = UVU^T$. $f(\mathbf{w})$ (same as $f(W)$ since $\mathbf{w}$ is nothing but reshaped W) may now be re-written as

$$\begin{aligned} f(\mathbf{w}) &= ||LX||_F^2 = \sum_{k=1}^{d} ||Lx^{(k)}||^2 = \sum_{k=1}^{d} ||UVU^T x^{(k)}||^2 \\ &= \sum_{k=1}^{d} ||UVy^{(k)}||^2 = \sum_{k=1}^{d} ||UY^{(k)}\mathbf{w}||^2 = ||M\mathbf{w}||^2 \end{aligned}$$

where $M^T = \left[Y^{(1)}U^T, \ldots, Y^{(d)}U^T\right]$ and $||M||_F = (\sum_{ij} M_{ij}^2)^{\frac{1}{2}}$ is the Frobenius norm. The resulting optimization problem for hard graphs is shown below.

$$\min_{\mathbf{w}} f(\mathbf{w}) = ||M\mathbf{w}||^2, \text{ s.t. } d_i \geq 1, i = 1, \ldots, n \tag{2}$$

The problem shown in (2) is a quadratic program in $\binom{n}{2}$ variables (length of $\mathbf{w}$). This is because for the set of n nodes, a total of $\binom{n}{2}$ (undirected) edges are possible. Optimizing (2) for all these variables at once may be computationally infeasible (Daitch et al., 2009). Hence, an incremental (greedy) algorithm is presented in (Daitch et al., 2009). The idea is solve the quadratic program (2) initially for a small subset of edges and then incrementally new edges to the problem which would improve the hard graph, and in the process remove any edge whose weight has been set to zero. This process is repeated until the graph can't be improved any further. New edges to be added to the program are determined based on violation of KKT conditions (Nocedal and Wright, 1999) of the Lagrangian.

Some more details of this incremental algorithm (e.g. how to choose the initial set of edges etc.) would have helped in improving our understanding of it (Daitch et al., 2009).

[1] equivalent to the Matlab operation: $\mathbf{w} = \texttt{reshape}(W > 0, 1, m)$

5.2 α-Soft Graphs

Sometimes it may be necessary to relax the degree constraint for some nodes. For example, we may want to have sparser connectivity for outlier nodes compared to nodes in high density regions. With this in mind, the degree constraint in (1) may be relaxed to the following

$$\sum_i (\max(0, 1 - d_i))^2 \leq \alpha n \tag{3}$$

where α is a hyper-parameter. Graphs estimated with this new constraint will be called *α-soft graphs*. If all nodes satisfied the hard degree requirement ($d_i \geq 1$ i.e. $1 - d_i \leq 0$, $\forall i$), then constraint (3) is trivially satisfied. Each *hinge loss* term, $\max(0, 1 - d_i)$, in the left hand side of (3) incurs a positive penalty when the corresponding node $\mathbf{x}_i$ violates the degree constraint i.e. $d_i < 1$. Note that the value of $f(\mathbf{w}) = \sum_i ||d_i \mathbf{x}_i - \sum_j \mathbf{w}_{ij} \mathbf{x}_j||^2$ is decreased (improved) by uniformly scaling down weights of all the edges. In that case, it is easy to verify that the constraint (3) will be satisfied with equality. If we define $\eta(\mathbf{w}) = \sum_i (\max(0, 1 - d_i))^2$, then

$$\eta(\mathbf{w}^*) = \alpha n \tag{4}$$

where $\mathbf{w}^*$ is a minimizer of $f(\mathbf{w})$ subject to constraint (3). Instead of incorporating constraint (3) directly, the algorithm for α-soft graphs in (Daitch et al., 2009) incorporates the constraint as a regularization term and instead solves the optimization problem shown in (5).

$$\min_{\mathbf{w}} f(\mathbf{w}) + \mu \ \eta(\mathbf{w}), \text{ s.t. } \mathbf{w} \geq 0 \tag{5}$$

where μ is a hyper-parameter. Let us now look at the algorithm for α-graph construction presented in (Daitch et al., 2009), which minimizes (5). Note that the objective in (5) is not directly dependent on α. We will see how that connection is established.

Let $\mathbf{w}$ be a solution to (5) for some given μ. In that case, following condition (4), $\mathbf{w}$ is also a solution for an α'-graph where $\alpha' = \frac{\eta(\mathbf{w})}{n}$. Additionally, as μ is increased, α' decreases accordingly. This is because, as μ is increased, the degree violation penalty, $\eta(\mathbf{w})$, is going to be reduced as the objective in (5) is minimized. Hence, the algorithm starts with an initial guess for μ which is then adjusted accordingly until α' comes close to α (the desired value), subject to some tolerance. In this iterative algorithm, the problem in (5) is solved once for each value of μ.

5.3 Properties of the Graphs

The hard and α-soft graphs may not be unique in general (Daitch et al., 2009). A few other properties of the hard and α-soft graphs are proved as well in (Daitch et al., 2009), which are reproduced below.

Theorem 3.1. For every $\alpha > 0$, every set of n vectors in $\mathbb{R}^d$ has a hard and an α-soft graph with at most $(d + 1)n$ edges.

This theorem suggests that the average degree ($\frac{2m}{n}$) of any node in the graph is at most $2(d+1)$. For situations where $n \gg d$, this theorem guarantees a sparse graph. Empirical validation of this is also presented in (Daitch et al., 2009). However, this bound will turn out to be very loose for high dimensional data e.g. data from Natural Language Processing where instances often have millions

of dimensions (or features). Daitch et al. (2009) also suggest that the average degree of a hard or α-soft graph of a set of vectors is a measure of the effective dimensionality of those vectors, and that it will be small if they lie close to a low-dimensional manifold of low curvature. Currently, no theoretical (or empirical) validation of this claim is presented in (Daitch et al., 2009). However, if it were indeed the case, then such graph construction process may be independently useful in discovering dimensionality of a set of points in high dimensional space. This may also suggest whether methods like Manifold Regularization (Belkin et al., 2005) is going to be effective on such set of points.

Daitch et al. (2009) prove an additional theorem about planarity of induced hard and α-soft ($\alpha > 0$) graphs in $\mathbb{R}^2$.

5.4 Results

A variety of experimental results are presented in (Daitch et al., 2009) which demonstrate properties of the constructed hard and 0.1-soft graphs ($\alpha = 0.1$) as well as their effectiveness in classification, regressions and clustering tasks. From Table 1 of (Daitch et al., 2009), it is interesting to note that the average degree of a node in the hard and soft graphs are lower than the upper bound predicted by Theorem 3.1. From the same table, we observe that hard and soft graph construction time increases sharply as n increases, which raises scalability concerns for such methods.

Once the graphs are constructed, the label inference algorithm in (Zhu et al., 2003) used for classification and regressions experiments. Performance of hard and soft graphs are compared against k-NN and ϵ-neighborhood graphs. From the results in Tables 2 and 3 (Daitch et al., 2009), we note that inference on hard and 0.1-soft graphs outperform these other graph construction methods.

5.5 Relationship to LNP (Wang and Zhang, 2008)

The Linear Neighborhood Propagation (LNP) algorithm (Wang and Zhang, 2008) is a two step process whose step 1 involves a graph construction step. In this case, the graph is constructed by minimizing an objective (6) which is similar to the LLE objective (Roweis and Saul, 2000). As in Section 5.1, the difference with LLE is that in LNP, the edge weight matrix W is symmetric and that all edge weights are non-negative (these constraints are not shown in (6)).

$$\min_W \sum_i ||\mathbf{x}_i - \sum_j W_{ij}\mathbf{x}_j||^2, \text{ s.t. } d_i = \sum_j W_{ij} = 1, i = 1, \ldots, n \tag{6}$$

By comparing (6) with (1) from Section 5.1, we note that the hard graph optimization enforces the constraint $d_i \geq 1$ while the LNP graph construction optimization (6) enforces $d_i = 1$. In other words, LNP is searching a smaller space of graphs compared to hard graphs.

6 Graph Kernels by Spectral Transform (Zhu et al., 2005)

Graph based transductive methods often try to strike a balance between *accuracy* (of the labeled nodes) and *smoothness* with respect to variation of labels over the graph. Smoothness enforces

the constraint that labels across a high weighted edge should not differ much. A central quantity in determining smoothness is the graph Laplacian, a matrix computed from edge weights. In (Zhu et al., 2005), it is noted that the eigenvectors of the graph Laplacian play a critical role in the resulting smoothness and that the eigenvectors can be combined using different weightings resulting in varying levels of smoothness. In (Zhu et al., 2005), such weightings of the eigenvectors of the graph Laplacian are called *spectral transforms*. A spectral transformation based method for estimating a graph kernel is presented in (Zhu et al., 2005), which will be discussed in this section. We first start with a review of the basic properties of graph Laplacian and its relationship to smoothness.

6.1 Graph Laplacian

Unnormalized Laplacian, L, of a graph $G = (V, E)$ is defined as

$$L = D - W$$

where W is the symmetric edge weight matrix, and D is a diagonal matrix where $D_{ii} = \sum_j W_{ij}$ and $D_{ij} = 0$ for $i \neq j$. We note that L, D, and W are all symmetric $n \times n$ matrices. Given a function, $f : V \rightarrow R$, which assigns scores (for a particular label) to each vertex in V, it is easy to verify that

$$f^T L f = \sum_{i,j \in V} W_{ij}(f(i) - f(j))^2 \tag{7}$$

We note that minimization of Equation 7 enforces the smoothness condition as in such a minimization, we shall end up with $f(i) \approx f(j)$ for edges with high W_{ij}.

We know that $W_{ij} \geq 0,\ \forall i, j$ and hence $f^T L f \geq 0,\ \forall f$ which implies that L is a positive semi-definite matrix. Let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and $\Phi_1, \Phi_2, \ldots \Phi_n$ be the complete set of eigenvalues and orthonormal eigenvectors of L. Smoothness penalty incurred by a particular eigenvector, Φ_i is then given by

$$\Phi_i^T L \Phi_i = \lambda_i \Phi_i^T \Phi_i = \lambda_i$$

Thus, as noted in (Zhu et al., 2005), eigenvectors with smaller eigenvalues are smoother. For illustration, a simple graph (with two segments) and its spectral decomposition are shown in Figure 3. From this figure we observe the following,

- $\lambda_1 = \lambda_2 = 0$, which is equal to the number of connected components in the graph (in this case 2). In general, A graph has k connected components iff $\lambda_i = 0$ for $i = 1, \ldots, k$. Moreover, the corresponding eigenvectors are constant within the connected components, which we observe in the first two plots of Figure 3 (b).
- The eigenvectors become more irregular with increasing eigenvalues.

This association between smoothness and ranked eigenvalues (and corresponding eigenvectors)will be exploited for learning smooth kernels over the graph is explained in next section.

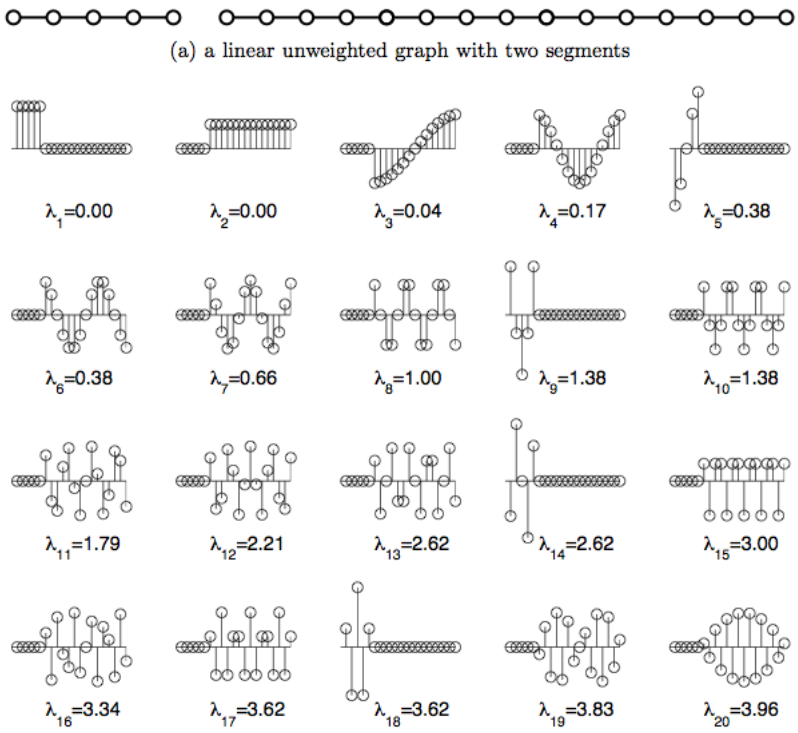

Figure 3: A linear-chain graph (part (a)) with two segments and its spectral decomposition (part (b)). Note that the eigenvectors are smoother for lower eigenvalues (Zhu et al., 2005).

6.2 Kernels by Spectral Transforms

In order to learn a kernel $K \in \mathbb{R}^{n \times n}$ which penalizes functions which are not smooth over the graph, the following kernel form is considered in (Zhu et al., 2005)

$$K = \sum_i \mu_i \phi_i \phi_i^T \tag{8}$$

where ϕ_is are the eigenvectors of the graph Laplacian L, and $\mu_i \geq 0$ are the eigenvalues of K. We note that K is the non-negative sum of outer products and hence a positive semi-definite matrix and hence a valid kernel matrix. We wold like to point out that in order to estimate K, an initial graph structure is necessary as estimation of K is dependent on L which is the Laplacian of a fixed graph.

In order to assign higher weight (μ_i) to outer product $\phi_i \phi_i^T$ of smoother eigenvector ϕ_i (and correspondingly smaller eigenvalue λ_i) of L, a *spectral transformation* function $r : \mathbb{R}_+ \to \mathbb{R}_+$ is defined, where r is non-negative and decreasing. By setting $\mu_i = r(\lambda_i)$, we rewrite Equation 8 as

follows

$$K = \sum_i r(\lambda_i)\phi_i\phi_i^T \tag{9}$$

The spectral transformation essentially reverses the order of the eigenvalues. Construction of kernels from graph Laplacians have also been the focus of some previous research (Chapelle et al., 2002),(Smola and Kondor, 2003). For example, setting $r(\lambda_i) = \frac{1}{\lambda_i+\epsilon}$ (where ϵ is a hyper-parameter), results in the Gaussian field kernel used in (Zhu et al., 2003).

One remaining crucial question is the choice of function r. Although there are many natural choices, it is not clear up front which one is most relevant for the current learning task. Moreover, tuning any hyper-parameter associated with any parametric form of r is another issue. In (Zhu et al., 2005), these issues are handled by learning a spectral transformation that optimizes kernel alignment to the labeled data [2]. This is described in the next section.

6.3 Optimizing Kernel Alignment

Let K_{tr} be the $l \times l$ sub-matrix of the full kernel matrix K, with K_{tr} corresponding to the l labeled instances $x_1, x_2, \ldots, x_l$. Let the $l \times l$ matrix T derived from the labels $y_1, y_2, \ldots, y_l$ be defined as: $T_{ij} = 1$ if $y_i = y_j$ and -1 otherwise. Empirical kernel alignment (Cristianini et al., 2001), (Lanckriet et al., 2004) between K_{tr} and T is defined as

$$A(K_{tr}, T) = \frac{\langle K_{tr}, T\rangle_F}{\sqrt{\langle K_{tr}, K_{tr}\rangle_F \langle T, T\rangle_F}} \tag{10}$$

where $< \cdot, \cdot >_F$ is the Frobenius product[3]. Please note that $A(K_{tr}, T)$ is maximized when $K_{tr} \propto T$. In (Zhu et al., 2005), the kernel matrix K (as given by Equation 8) is estimated by maximizing the kernel alignment objective in Equation 10, with the optimization defined directly over the transformed eigenvalues μ_i, without any parametric assumption for r which would have otherwise linked μ_i with the corresponding eigenvalue λ_i of L. Since the optimization is directly over the variables μ_i's, we need a way to encode the constraint that μ_i with lower index i should have higher value (Section 6). In (Zhu et al., 2005), following *order constraints* are added to the optimization to achieve this goal

$$\mu_i \geq \mu_{i+1},\ i = 1, 2, \ldots, n-1$$

The estimated kernel matrix K should be a positive semi-definite matrix and hence the kernel alignment based optimization described above can be set up as a Semi-Definite Program (SDP). However, SDP optimization suffers from high computation complexity (Boyd and Vandenberghe, 2004). Fortunately, the optimization problem can also be set up in a more computationally efficient form. The resulting optimization problem is presented below

$$\begin{array}{c} \max_K \quad A(K_{tr}, T) \\ \text{subject to} \quad K = \sum_i \mu_i \phi_i \phi_i^T \\ \mathrm{Tr}(K) = 1 \end{array}$$

[2] Please remember that the goal in (Zhu et al., 2005) is to learn a kernel in the (transductive) semi-supervised setting and so a few instances are labeled.

[3] $\langle M, N\rangle_F = \sum_{ij} M_{ij} N_{ij}$

$$\mu_i \geq 0,\ i = 1, 2, \ldots, n$$
$$\mu_i \geq \mu_{i+1},\ i = 1, 2, \ldots, n-1$$

The trace constraint is needed to fix the scale invariance of kernel alignment (Zhu et al., 2005). The problem can be equivalently rewritten as

$$\max_K \quad \langle K_{tr}, T \rangle_F \tag{11}$$
$$\text{subject to} \quad K = \textstyle\sum_i \mu_i \phi_i \phi_i^T \tag{12}$$
$$\langle K_{tr}, K_{tr} \rangle_F \leq 1 \tag{13}$$
$$\mu_i \geq 0,\ i = 1, 2, \ldots, n \tag{14}$$
$$\mu_i \geq \mu_{i+1},\ i = 1, 2, \ldots, n-1 \tag{15}$$

The kernel obtained by solving the above optimization is called *Order Constrained Kernel* (Zhu et al., 2005), which we shall denote by K_{OC}. Another property of the Laplacian is exploited in (Zhu et al., 2005) to define an *Improved Order Constrained Kernel*, which we shall refer to as K_{IOC}. In particular, $\lambda_1 = 0$ for any graph Laplacian (Chung, 1997). Moreover, if there are *no disjoint components* in the graph then ϕ_1 is constant overall[4]. In that case, $\phi_1 \phi_1^T$ is a constant matrix which does not depend on L, with the product $\mu_1 \phi_1 \phi_1^T$ acting as a bias term in the definition of K.

In that case constraint (15) from above can be replaced with

$$\mu_i \geq \mu_{i+1},\ i = 1, 2, \ldots, n-1, \text{ and } \phi_i \text{ not constant overall} \tag{16}$$

We would like to point out that constraint (16) is meant to affect only μ_1 and that too in connected graphs only, as in all other cases ϕ_i is not going to be constant overall. Replacing constraint (15) with (16) and solving the resulting optimization results in K_{IOC}.

6.4 Results

Results from a large number of experiments comparing K_{OC} and K_{IOC} to six other standard kernels: Gaussian (Zhu et al., 2003), Diffusion (Smola and Kondor, 2003), Max-Align (Lanckriet et al., 2004), Radial Basis Function (RBF), Linear and Quadratic. Out of these, RBF, Linear and Quadratic are unsupervised while the rest (including the proposed order kernels) are semi-supervised in nature. All 8 kernels are combined with the same Support Vector Machine (SVM) (Burges, 1998). Accuracy of the SVMs on unlabeled data is used as the evaluation metric. Experimental results from 7 different datasets are reported. For each dataset, an unweighted 10-NN (with the exception of one 100-NN) graph is constructed using either Euclidean or cosine similarity. In all cases, the graphs are connected and hence constraint (16) is used while estimating K_{IOC}. Smallest 200 eigenvectors of Laplacians are used in all the experiments. For each dataset, 5 training set sizes are used and for each set 30 random trials are performed. From the experimental results presented in (Zhu et al., 2005), we observe that the improved order constrained kernel, K_{IOC}, outperformed all other kernels.

[4]Otherwise, if a graph has k connected components, then the first k eigenvectors are piecewise constant over the components.

6.5 Discussion

In the sections above, we reviewed the kernel-alignment based spectral kernel design method described in (Zhu et al., 2005). At one end, we have the *maximum-alignment* kernels (Lanckriet et al., 2004) and at the other end we have the parametric Gaussian field kernels (Zhu et al., 2003). The order-constrained kernels introduced in (Zhu et al., 2005) strikes a balance between these two extremes. In such spectral transformation based methods, it will be interesting to see how the resulting edge weights (W) vary with different spectral transformations.

6.6 Relationship to (Johnson and Zhang, 2008)

In (Zhu et al., 2005) and its review in the sections above, we have seen how to derive a kernel from the graph Laplacian and effectiveness of the derived kernel for supervised learning. Johnson and Zhang (2008) look at the related issue of the relationship between graph-based semi-supervised learning and supervised learning. Let k be a given a kernel function and K the corresponding kernel matrix. Let $L = K^{-1}$ be the Laplacian of a graph. In that case, in Theorem 3.1 of (Johnson and Zhang, 2008), it is proved that graph-based semi-supervised learning using Laplacian $L = K^{-1}$ is the equivalent to a kernel-based supervised learner (e.g. SVM) with kernel function k.

In (Zhu et al., 2005), spectrum of the Laplacian of a fixed graph is modified to compute a kernel. While in (Johnson and Zhang, 2008), the kernel is created up front using all the data and without need for any a-priori fixed graph structure.

7 Discussion & Future Work

In this section we further analyze the methods reviewed in this report and suggest steps for future work.

- **Degenerate Solutions**: Graph-based SSL methods may generate degenerate (or trivial) solutions sometimes i.e. solutions where all nodes in the graph are assigned the same label. In order to handle this issue, various heuristics e.g. Class Mass Normalization (CMN) (Zhu et al., 2003) have been proposed. In CMN, class proportion information is taken into account account and the unlabeled points are classified so that the output class proportions are similar to the input class proportion. This is done by post-processing the output obtained from the graph-based SSL method.

  From Section 6.1, we know that the eigenvectors corresponding to the smaller eigenvalues are smoother and in particular cases constant across each connected component. It will be quite insightful to explore the relationship between spectrum of the Laplacian and the possibility for degenerate solutions in a given graph. For a given algorithm, it will be useful to know the types of graphs on which it is likely to produce degenerate solutions so that more principled corrective steps can be taken accordingly. Transformation of the Laplacian may be one such possibility.

- **Regular vs Irregular Graphs**: Jebara et al. (2009) emphasize the importance of regular graphs (all nodes have same degree) for graph-based SSL compared to irregular graphs often

generated by popular graph-construction methods such as k-NN. It is not directly clear what effect node degrees have on the resulting classification. In *under-regularized* methods (Zhu et al., 2003), it is conceivable that presence of a high degree node may have adverse effect. In such cases, the algorithm may end up assigning most of the nodes the same label as the high degree node, thereby producing a degenerate solution as discussed previously. Recent methods have tried to address this problem by decreasing importance of high-degree nodes through additional regularization (Baluja et al., 2008; Talukdar and Crammer, 2009). In the light of these developments, it will be interesting to see whether the need for regular graphs can be substituted by applying these new regularized methods on corresponding irregular graphs.

- **Construction based on Global Property**: All the graph construction methods reviewed so far emphasize on local properties of the graph e.g. each node should have certain number of neighbors (either exactly or approximately). There seems to be a lot more room for exploration in terms of enforcing other node local properties or even global properties (e.g. diameter of the graph). Enforcing global property directly may lead to combinatorial explosion and so certain efficiently enforceable approximation of the global property may have to considered. Moreover, all induced graphs are undirected in nature. This is primarily because graph-based transductive methods are targeted towards undirected graphs. However, transductive methods for directed graphs exist (Zhou et al., 2005) and so there is scope for induction of graphs with non-symmetric edge weight matrices.

- **Alternative Graph Structures**: All graph structures considered so far have instance-instance edges. There has been very little work in learning other types of graphs e.g. hybrid graphs with both features and instances present in them. Such types of graphs are used in (Wang, Z. and Song, Y. and Zhang, C., 2009), though the graph structure in this case is fixed a-priori and is not learned.

- **Including Prior Information**: All the graph construction methods surveyed so far (Jebara et al., 2009; Daitch et al., 2009; Wang and Zhang, 2008) use the locally linear reconstruction (LLR) principle (Roweis and Saul, 2000) in some way or the other. For example, (Daitch et al., 2009; Wang and Zhang, 2008) use variants of LLR during construction of the graph itself, while LLR is used for edge re-weighting in (Jebara et al., 2009). This suggests that there is room for exploration in this space in terms of other construction heuristics and methods. For example, none of these methods take prior knowledge into account. Sometimes we may know a-priori existence of certain edges and in the resulting graph we would like to include those edges. Such partial information may regularize the construction method and result in more appropriate graphs for the learning task at hand.

- **Including Labeled Data**: Unlike the empirical kernel alignment step in (Zhu et al., 2005), none of the graph construction methods (Jebara et al., 2009; Daitch et al., 2009; Wang and Zhang, 2008) exploit available labeled data during construction of the graph. This is related to the comment above about incorporation of prior-information. Labeled data may be seen as one type of prior information which could be useful in customizing the graph construction for the current learning task.

- **New Application Settings**: Most of the experimental datasets considered in the papers surveyed in this report have small number of instances and often small number of dimensions.

Since the methods reviewed can incorporate unlabeled data and acquiring unlabeled instances in most domains is often not hard, the number of instances input into the algorithms can be readily increased. Also, in many domains (e.g. Natural Language Processing), the number of dimensions (features) can run up to millions. It will be interesting to see effectiveness of these methods in such settings.

8 Conclusion

In this report, we have reviewed several recently proposed methods for graph-construction (Wang and Zhang, 2008; Jebara et al., 2009; Daitch et al., 2009). Locally Linear Reconstruction (Roweis and Saul, 2000) based edge weighting and enforcement of degree constraint on each node is a common theme in these methods. Daitch et al. (2009) further analyze properties of the induced graphs, while Jebara et al. (2009) emphasize the need for regular graphs in graph-based SSL. Additionally, we also looked at semi-supervised spectral kernel design (Zhu et al., 2005). Even though the graph structure is fixed a-priori in this case, this method highlights useful properties of the graph Laplacian and also demonstrates the relationship between the graph structure and a kernel that can be derived from it. Finally, we suggested a few steps for future work on graph construction for graph-based SSL.

References

S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. *Proceedings of WWW-2008*, 2008.

M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In *Proc. of Conference on AI and Statistics*, 2005.

Y. Bengio, O. Delalleau, and NL Roux. Semi-Supervised Learning, chapter Label Propogation and Quadratic Criterion, 2007.

J.L. Bentley. Multidimensional divide-and-conquer. *Commun. ACM*, 23:214–229, 1980.

A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104. ACM New York, NY, USA, 2006.

S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

O. Chapelle, J. Weston, and B. Scholkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems*, 2002.

O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning*. MIT press, 2006.

F.R.K. Chung. *Spectral graph theory*. Amer Mathematical Society, 1997.

N. Cristianini, J. Kandola, and A. Elissee. On kernel target alignment. *Advances in Neural Information Processing Systems 14*, 2001.

S.I. Daitch, J.A. Kelner, and D.A. Spielman. Fitting a graph to vector data. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM New York, NY, USA, 2009.

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

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*. ACM New York, NY, USA, 2009.

Rie Johnson and Tong Zhang. Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, 54(1):275–288, 2008.

J. Lafferty and L. Wasserman. Statistical analysis of semi-supervised regression. *Advances in Neural Information Processing Systems*, 20:801–808, 2007.

G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

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

P. Niyogi. Manifold regularization and semi-supervised learning: Some theoretical analyses. Technical report, Technical Report TR-2008-01, Computer Science Department, University of Chicago, 2008.

J. Nocedal and S.J. Wright. *Numerical optimization*. Springer, 1999.

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

A. Singh, R.D. Nowak, and X. Zhu. Unlabeled data: Now it helps, now it doesnt. In *Advances in Neural Information Processing Systems*, 2009.

A.J. Smola and R. Kondor. Kernels and regularization on graphs. In *Learning theory and Kernel machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003: proceedings*, page 144. Springer Verlag, 2003.

A. Subramanya and J. Bilmes. Soft-Supervised Learning for Text Classification. In *Proceedings of EMNLP. Honolulu, Hawaii*, 2008.

Partha Pratim Talukdar and Koby Crammer. New regularized algorithms for transductive learning. In *ECML-PKDD*, 2009.

P.P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. Pereira. Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 581–589, 2008.

K. Tsuda, H.J. Shin, and B. Scholkopf. Fast protein classification with multiple networks. *Bioinformatics*, 21(90002), 2005.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

F. Wang and C. Zhang. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):55–67, 2008.

J. Wang, T. Jebara, and S.F. Chang. Graph transduction via alternating minimization. In *Proceedings of the 25th international conference on Machine learning*, pages 1144–1151. ACM New York, NY, USA, 2008.

Wang, Z. and Song, Y. and Zhang, C. Knowledge Transfer on Hybrid Graph. In *IJCAI*, 2009.

D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, page 321. The MIT Press, 2004.

D. Zhou, J. Huang, and B. Scholkopf. Learning from labeled and unlabeled data on a directed graph. In *ICML*, 2005.

X. Zhu. Semi-supervised learning literature survey. 2005.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *ICML-03, 20th International Conference on Machine Learning*, 2003.

X. Zhu, J. Kandola, J. Lafferty, and Z. Ghahramani. Graph kernels by spectral transforms. In *Semi-Supervised Learning*. MIT Press, 2005.