Fast Multilevel Transduction on Graphs

Fei Wang\textsuperscript{*}  Changshui Zhang\textsuperscript{†}

Abstract

The recent years have witnessed a surge of interest in graph-based semi-supervised learning methods. The common denominator of these methods is that the data are represented by the nodes of a graph, the edges of which encode the pairwise similarities of the data. Despite the theoretical and empirical success, these methods have one major bottleneck which is the high computational complexity (since they usually require the computation of matrix inverse). In this paper, we propose a multilevel scheme for speeding up the traditional graph-based semi-supervised learning methods. Unlike other accelerating approaches based on pure mathematical derivations, our method has explicit physical meanings with some graph intuitions. We also analyze the relationship of our method with multigrid methods, and provide a theoretical guarantee of the performance of our method. Finally the experimental results are presented to show the effectiveness of our method.

1 Introduction

In many practical applications of pattern classification and machine learning, one often faces a lack of sufficient labeled data, since labeling often requires expensive human labor. However, in many cases, large numbers of unlabeled data can be far easier to obtain. For example, in text classification, one may have an easy access to a large database of texts by crawling the web, but only a small part of them are classified by hand. Therefore, the problem of effectively combining unlabeled data with labeled data is of central importance in machine learning.

Semi-supervised learning (SSL) methods are just the kind of methods which aims to learn from partially labeled data [5]. The key to SSL problems is the cluster assumption [14]: (1) nearby points are likely to have the same label (local consistency); (2) points on the same structure (such as a cluster or a submanifold) are likely to have the same label (global consistency).

It is straightforward to associate cluster assumption with the manifold analysis methods developed in recent years [2, 9, 10]. These methods first assume that the data points (nearly) reside on a low-dimensional manifold (which is called manifold assumption in [5]), and then try to discover such manifold by preserving some local structure of the dataset. It is well known that graphs can be viewed as discretizations of manifolds [1], consequently, numerous graph based SSL methods have been proposed in recent years, and graph based SSL has been becoming one of the most active research area in semi-supervised learning community [5].

However, in spite of the intensive study of graph based SSL methods, there are still some open issues which have not been addressed properly, such as:

1. How to select an appropriate similarity measure between pairwise data automatically;
2. How to speed up these algorithms for handling large-scale dataset (since they usually require the computation of matrix inverse).

In our previous paper, we have proposed a good way to address the first issue [12]. In this paper we will be mainly concentrate on the accelerating issue.

As another research field, the psychologists have long studied the way that how people perceive the world, among which the Gestalt psychology [13] is a movement that began just prior to World War I. It made important contributions to the study of visual perception and problem solving. The Gestalt approach emphasizes that we perceive objects as well-organized patterns rather than separate component parts. According to this approach, when we open our eyes we do not see fractional particles in disorder. Instead, we notice larger areas with defined shapes and patterns. The “whole” that we see is something that is more structured and cohesive than a group of separate particles. Clearly, the focal point of Gestalt theory is the idea of grouping, i.e., how we tend to interpret a visual field or problem in a certain way. The main factors that determine grouping including proximity, similarity and simplicity.

Inspired by the Gestalt laws, we propose a fast multilevel graph learning algorithm. In our method, the data graph is first coarsened level by level based on the similarity between pairwise data points, then the learning procedure can be performed on a graph with a much small size. Finally the solution on the coarsened graph will be refined back level by level to get the solution of the initial problem. Our experimental

\textsuperscript{*}Department of Automation, Tsinghua University.
\textsuperscript{†}Department of Automation, Tsinghua University.
results show that this strategy can improve the speed of graph-based SSL algorithms significantly. And we also give a theoretical guarantee on the performance of our algorithm.

The rest of this paper is organized as follows. Some notations and related works will be introduced in section 2. We will formally present the new algorithm in section 3 and analyze its connection with multigrid methods in section 4. The experimental results will be shown in section 5, followed by the conclusions in section 6.

2 Notations and Related Works

We suppose that there are a set of data points $X = \{x_1, \cdots, x_i, \cdots, x_{i+u}\}$, of which $X_l = \{x_1, x_2, \cdots, x_l\}$ are labeled as $y_i \in \mathcal{L}$ ($1 < i \leq l$), $\mathcal{L} = \{1, 2, \cdots, C\}$ is the label set) and the remaining points $X_u = \{x_{i+1}, \cdots, x_{i+u}\}$ are unlabeled. The goal is to predict the labels of $X_u$.

Generally, the graph based SSL methods [1, 14, 16] first model the whole dataset as a weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where node set $\mathcal{V}$ corresponds to the dataset $X = X_l \cup X_u$, and $\mathcal{E}$ is the edge set. Associate with each edge $e_{ij} \in \mathcal{E}$ is a weight $w_{ij}$, which is usually computed by

$$w_{ij} = \exp \left(-\beta \|x_i - x_j\|^2 \right) ,$$

where $\beta$ is a free parameter which is usually set empirically. Many methods have been proposed to determine $\beta$ automatically [5], however, there is no reliable approach [14].

In Zhou’s terminology [14], the classification result on $X$ can be represented by an $n \times C$ matrix $F$ with nonnegative entries (here $n = l + u$). And $x_i$’s label $t_i$ is determined by

$$t_i = \arg\max_{c \in \mathcal{C}} F_{ij}.$$ 

$F_{ij}$ is the $(i, j)$-th entry of $F$. Therefore $F_{ij}$ can be treated as a measure of possibility that $x_i$ belongs to class $j$ (If we further normalize the rows of $F$ so that $\sum_j F_{ij} = 1, \forall 1 \leq i \leq n$, then $F_{ij}$ can be regarded as the probability of $x_i$ belonging to class $j$). Based on this point of view, we will call $F$, which is the $c$-th column of $F$, the classification vector of the $c$-th class. Similarly, we will call $y_x$ the original label vector of class $c$ with its $i$-th entry

$$y_{x_i}^c = \begin{cases} 1 & \text{if } x_i \text{ is labeled as } c \\ 0 & \text{if } x_i \text{ is unlabeled or its label is not } c \end{cases}$$

For notational convenience, we will omit the superscripts, and use $f$ and $y$ to denote an arbitrary classification vector and its corresponding original label vector.

A common principal that for graph based SSL is to minimize the following cost function with some constraints

$$\mathcal{J}(f) = f^T S f + \gamma \|f - y\|^2 ,$$

where the first term is the smoothness term, which measures the total variation of the data labels with respect to the intrinsic structure, and $S$ is a smoothness matrix [1]; the second term is the fit term, which measures how well the predicted labels fit the original labels, and $\gamma > 0$ is the regularization parameter.

Table 1 lists some recently proposed methods and their corresponding parameters. The optimal $f$ can be solved by set $\frac{\partial \mathcal{J}(f)}{\partial f} = 0$, which results in $f = (S + \gamma I)^{-1} y$. Obviously, the computation of $(S + \gamma I)^{-1}$ is impractical for large datasets.

3 Multilevel Semi-Supervised Learning on Graphs

In this section we will introduce a new graph based SSL algorithm that can (1) determine the weight on each edge of the graph automatically; (2) handle large scale datasets in a multilevel way.

Table 1: Popular graph based semi-supervised learning objectives and their corresponding parameters.

<table>
<thead>
<tr>
<th>Method</th>
<th>$S$</th>
<th>$\gamma$</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph Mincuts [4]</td>
<td>$L = D - W$</td>
<td>$\infty$</td>
<td>$f_i \in {0, 1}$</td>
</tr>
<tr>
<td>Label Propagation [16]</td>
<td>$L = D - W$</td>
<td>$\infty$</td>
<td>None</td>
</tr>
<tr>
<td>Gaussian Random Fields [16]</td>
<td>$L = D - W$</td>
<td>$\infty$</td>
<td>None</td>
</tr>
<tr>
<td>Local and Global Consistency [14]</td>
<td>$L = I - D^{-1/2}WD^{-1/2}$</td>
<td>$\gamma &gt; 0$</td>
<td>None</td>
</tr>
<tr>
<td>Tikhonov Regularization [1]</td>
<td>$L^p$, $p \in \mathbb{N}$ or $\exp(-tL)$, $t \in \mathbb{R}$</td>
<td>$1/t$</td>
<td>None</td>
</tr>
<tr>
<td>Interpolated Regularization[1]</td>
<td>$L^p$, $p \in \mathbb{N}$ or $\exp(-tL)$, $t \in \mathbb{R}$</td>
<td>$\infty$</td>
<td>$\sum_i f_i = 0$</td>
</tr>
<tr>
<td>Linear Neighborhood Propagation[12]</td>
<td>$L_u = I - W_u$</td>
<td>$\gamma &gt; 0$</td>
<td>None</td>
</tr>
</tbody>
</table>
3.1 Graph Construction  
Unlike some traditional methods which construct a fully connected graph [14, 16], our strategy here is to construct a connected neighborhood graph as in [3]. This can save some storage requirement and make the final problem sparse. More concretely, the neighborhood system for $X$ is defined as

**Definition 3.1. (Neighborhood System)** Let $\mathcal{N} = \{ \mathcal{N}_i \mid \forall x_i \in X \}$ be a neighborhood system for $X$, where $\mathcal{N}_i$ is the neighborhood of $x_i$. Then $\mathcal{N}_i$ satisfies: (1) $x_i \notin \mathcal{N}_i$ (self-exclusion); (2) $x_i \in \mathcal{N}_j \Leftrightarrow x_j \in \mathcal{N}_i$ (symmetry).

In this paper, $\mathcal{N}_i$ is defined in the following way: $x_j \in \mathcal{N}_i$ iff $x_j \in \mathcal{K}_i$ or $x_j \in \mathcal{K}_j$, where $\mathcal{K}_i$ is the set that contains the $k$ nearest neighbors of $x_i$.

Based on the above definitions, we can construct the graph $G$ where there is an edge links nodes $x_i$ and $x_j$ if $x_j \in \mathcal{N}_i$. Thus we can define an $n \times n$ weight matrix $W$ for graph $G$, with its $(i, j)$-th entry

$W_{ij} = w_{ij} \begin{cases} > 0 & \text{if } x_j \in \mathcal{N}_i \\ = 0 & \text{otherwise} \end{cases}$.

Instead of considering pairwise relationships as Eq.(2.1) in traditional graph-based methods, we propose to incorporate the neighborhood information of each point to the estimation of $w_{ij}$. Similar to the method we proposed in [12], we assume each data can be linearly reconstructed from its $k$-nearest neighbors, thus we should minimize the reconstruction error

$\varepsilon_i = \left\| x_i - \sum_{j : x_j \in \mathcal{K}_i} \alpha_{ij} x_j \right\|^2$

$= \sum_{j, k : x_j, x_k \in \mathcal{K}_i} \alpha_{ij} \mathbf{G}_{jk}^i \alpha_{ik}$

for all $1 \leq i \leq n$. Here $\alpha_{ij}$ is the contribution of $x_j$ to $x_i$, subject to $\sum_{j \in \mathcal{K}_i} \alpha_{ij} = 1$, $\alpha_{ij} \geq 0$. And $\mathbf{G}_{jk}^i = (x_i - x_j)^T(x_i - x_k)$ is the $(j, k)$-th entry of the local Gram matrix at point $x_j$. Obviously, the more similar $x_j$ to $x_i$, the larger $\alpha_{ij}$ will be. Thus $\alpha_{ij}$ can be used to measure how similar $x_j$ to $x_i$. One issue should be addressed here is that usually $\alpha_{ij} \neq \alpha_{ji}$.

Thus the reconstruction weights of each data object can be solved by the following $n$ standard quadratic programming problems

$\begin{align*}
\min_{\alpha_{ij}} & \sum_{j, k : x_j, x_k \in \mathcal{K}(x_i)} \alpha_{ij} \mathbf{G}_{jk}^i \alpha_{ik} \\
\text{s.t.} & \sum_{j} \alpha_{ij} = 1, \; \alpha_{ij} \geq 0
\end{align*}$

However, we do not use $\alpha_{ij}$ as the similarity between $x_i$ and $x_j$ directly as in [12]. Recalling the definition of the neighborhood system in definition 1, we can compute
provides a graphical overview of our algorithm, and the
ing represents the level of graph scale), we first split
scheme is composed of three phases: (1) recursive level by level
details of the approach will be described in the following
Graphs
Below we will introduce a novel semi-supervised learning on graphs. The
scheme for semi-supervised learning on graphs. The
definitions will be introduced in the following subsections.
3.2 Multilevel Semi-Supervised Learning on Graphs
Below we will introduce a novel multilevel
scheme for semi-supervised learning on graphs. The
scheme is composed of three phases: (1) graph coarsening; (2) initial classification; (3) solution refining. Fig. 1 provides a graphical overview of our algorithm, and the
details of the approach will be described in the following
subsections.
3.2.1 Graph Coarsening We now present the recursive level by level graph coarsening phase in our method. In each coarsening step a new, approximately equivalent semi-supervised learning problem will be defined, with the number of vertices of the graph reducing to a fraction (typically around 1/2) of the original graph.
In the following we will describe the first coarsening step. Starting from graph \( G^0 = \mathcal{G} \) (the superscript represents the level of graph scale), we first split \( V^0 = \mathcal{V} \) into two sets, \( C^0 \) and \( F^0 \), subject to \( C^0 \cup F^0 = \mathcal{V} \), \( C^0 \cap F^0 = \Phi \). The set \( C^0 \) will be used as the node set of the coarser graph of the next level, i.e. \( V^1 = C^0 \).
And the nodes in \( C^0 \) are called \( C \)-nodes, which is defined as:

**Definition 3.2. (C-nodes and F-nodes) Given a graph \( G^l = (\mathcal{V}^l, \mathcal{E}^l) \), we split \( \mathcal{V}^l \) into two sets, \( C^l \) and \( F^l \) satisfying \( C^l \cup F^l = \mathcal{V}^l \), \( C^l \cap F^l = \Phi \), \( C^l = \mathcal{V}^{l+1} \). And each node in \( C^l \) must satisfy one of the following conditions:
(1) it is labeled;
(2) it strongly influences at least one node in \( F^l \) on level \( l \).
We will call the nodes in \( C^l \) \( C \)-nodes, and the nodes in \( F^l \) \( F \)-nodes.
Here strongly influence means

**Definition 3.3. (Strongly Influence) A node \( x_i \) strongly influences \( x_j \) on level \( l \) means that
\[
(3.7) \quad w_{ij}^l \geq \delta \sum_k w_{kj}^l
\]
where \( 0 < \delta < 1 \) is a control parameter, and \( w_{ij}^l \) is the weight of the edge linking \( x_i \) and \( x_j \) on \( G^l \).

Let \( f^0 = f \) be an classification vector we want to solve, and \( f^1 \) be its corresponding classification vector on \( G^1 \) (hence the dimensionality of \( f^1 \) should be equivalent to \( n^1 \), the cardinality of \( \mathcal{V}^1 \)). Without the loss of generality, we assume that \( f^0 \) can be linearly interpolated from \( f^1 \), that is
\[
(3.8) \quad f^0 = P^{[0,1]} f^1
\]
where \( P^{[0,1]} \) is the interpolation matrix of size \( n^0 \times n^1 \) \((n^0 = n)\), subject to \( \sum_{i} P_{ij}^{[0,1]} = 1 \). Therefore, bringing Eq.(3.8) into Eq.(2.3), we can get the coarsened problem which aims to minimize

\[
(3.9) \mathcal{J}(f^1) = f^T P^{[1,0]} \mathcal{S} P^{[0,1]} f^1 + \gamma \| P^{[0,1]} f^1 - y \|^2
\]
where \( P^{[1,0]} \) is the transpose of \( P^{[0,1]} \). In this paper we will use combinatorial Laplacian as the smoothness matrix because of its wide applicability \([1, 5, 16]\), i.e.
\[
(3.10) \quad S = D - W,
\]
where \( W \) is the weight matrix constructed in the manner introduced in section 3.1, and \( D \) is a diagonal matrix with its \((i,i)\)-th entry \( D_{ii} = \sum_j W_{ij} = \sum_j w_{ij}^0 \). Written in its element form,
\[
\mathcal{J}(f^0) = \sum_{i,j \in V^0} w_{ij}^0 (f_{ij}^0 - f_{ij}^0)^2 + \sum_{i \in V^0} (f_{i}^0 - y_i)^2
\]
\[
\mathcal{J}(f^1) = \sum_{i,j \in V^0} w_{ij}^1 \left( \sum_{k \in V^1} P^{[1,0]}_{jk} f_k^1 - \sum_{l \in V^1} P^{[0,1]}_{jl} f_{lj}^0 \right)^2
\]
\[
= \sum_{i \in V^0} \left( \sum_{k \in V^1} P^{[1,0]}_{ik} f_k^1 - y_i \right)^2
\]
\[
\mathcal{J}(f^0) = \sum_{i,j \in V^0} w_{ij}^0 (f_{ij}^0 - f_{ij}^0)^2 + \sum_{i \in V^0} (f_{i}^0 - y_i)^2
\]
\[
\mathcal{J}(f^1) = \sum_{i,j \in V^0} w_{ij}^1 \left( \sum_{k \in V^1} P^{[1,0]}_{jk} f_k^1 - \sum_{l \in V^1} P^{[0,1]}_{jl} f_{lj}^0 \right)^2
\]
\[
= \sum_{i \in V^0} \left( \sum_{k \in V^1} P^{[1,0]}_{ik} f_k^1 - y_i \right)^2
\]
where
\begin{equation}
(3.11) \quad w^1_{kl} = \frac{1}{2} \sum_{i,j} w^0_{ij} (P^{[0,1]}_{ji} - P^{[0,1]}_{il})(P^{[0,1]}_{jk} - P^{[0,1]}_{lk}).
\end{equation}

The first term of Eq.(3.11) can be considered as the smoothness of \( f^1 \) on graph \( G^1 \), and the weight on the edge linking \( x_k \) and \( x_l \) on \( G^1 \) can be computed by Eq.(3.11). Moreover, we can generalize the above conclusion and get the following theorem:

**Theorem 3.1.** The edge weights on graph \( G^{l+1} \) can be computed from the edge weights on \( G^l \) by
\begin{equation}
(3.12) \quad w^{l+1}_{uu} = \frac{1}{2} \sum_{i,j} w^l_{ij} (P^{[l+1]}_{iu} - P^{[l+1]}_{iv})(P^{[l+1]}_{iu} - P^{[l+1]}_{iv}).
\end{equation}

**Proof.** See Appendix 1.

An issue should be addressed here is that for computational efficiency, the above coarsening weight equation can be somewhat simplified to the following **Iterated Weighted Aggregation** strategy [17], which compute \( w^l_{uu} \) by
\begin{equation}
(3.13) \quad w^{l+1}_{uu} = \frac{1}{2} \sum_{i,j} P^{[l+1]}_{iu} w^l_{ij} P^{[l+1]}_{jv}.
\end{equation}

It can be shown that Eq.(3.13) can provide a good approximation to Eq.(3.12) in many cases [8].

Now the only remaining problem is how to determine the interpolation matrix, and we will leave it to section 3.2.3. To summarize, given a graph \( G^l = (V^l, E^l) \), we have to do three things for coarsening it to \( G^{l+1} = (V^{l+1}, E^{l+1}) \): (1) select the \( C \)-nodes from \( V^l \) based on definition 2, and let \( V^{l+1} = C^l \); (2) compute the interpolation matrix \( P^{[l+1]} \) (see section 3.2.3 for details); (3) compute the edge weights by Eq.(3.12). After these three steps, the graph will be coarsened to the next level.

### 3.2.2 Initial Classification

Assuming the data graph \( G \) has been coarsened recursively to some level \( s \), then the semi-supervised classification problem defined on \( G^s \) is to minimize
\[
J(f^s) = f^{sT} P^{[s,s-1]} \ldots P^{[1,0]} S f^s + \gamma \| P^{[0,1]} \ldots P^{[s-1,s]} f^s - y \|^2,
\]
where \( P^{[i,i-1]} = (P^{[i-1,i]})^T \), and \( S \) is defined in Eq.(3.10). Therefore, let \( \frac{\partial J(f^s)}{\partial f^s} = 0 \), then
\[
\begin{align*}
\frac{\partial J(f^s)}{\partial f^s} &= P^{[s,s-1]} \ldots P^{[1,0]} S P^{[0,1]} \ldots P^{[s-1,s]} f^s + \gamma P^{[s,s-1]} \ldots P^{[1,0]} \left( P^{[0,1]} \ldots P^{[s-1,s]} f^s - y \right) \\
&= (L^s) f^s - \gamma P^{[s,s-1]} \ldots P^{[1,0]} y = 0 \\
\implies f^s &= \gamma (L^s)^{-1} P^{[s,s-1]} \ldots P^{[1,0]} y.
\end{align*}
\]

Here \( I \) is the \( n \times n \) identity matrix. Moreover, we have the following theorem

**Theorem 3.2.** The matrix \( L^s = P^{[s,s-1]} \ldots P^{[1,0]} (S + \gamma I) P^{[0,1]} \ldots P^{[s-1,s]} \) is invertible.

**Proof.** For \( \forall a \neq 0 \), since \( w_{ij} \geq 0 \), \( \gamma > 0 \), then \( a^T (S + \gamma I) a = \sum_{i,j} w_{ij} (a_i - a_j)^2 + \gamma \sum_i a_i^2 > 0 \). Thus the matrix \( S + \gamma I \) is positive definite. If \( \text{rank} (P^{[0,1]}) = n \) \( (n^1 \) is the cardinality of \( V^1 \)), and let \( a \) be an arbitrary vector, then \( P^{[0,1]} a \neq 0 \). Let \( r = P^{[0,1]} a \), then we have
\begin{equation}
(3.14) \quad a^T P^{[1,0]} (S + \gamma I) P^{[0,1]} a = r^T (S + \gamma I) r > 0
\end{equation}

Therefore \( P^{[1,0]} (S + \gamma I) P^{[0,1]} \) is also positive definite. Similarly, if we can guarantee that \( \text{rank} (P^{[l,l-1]}) = n^l \), \( \forall 1 \leq l \leq s \) \( (n^l \) is the cardinality of \( V^l \)), then \( L^s \) will be invertible. Fortunately, the way we define the interpolation matrix (which will be introduced in the next subsection) can meet this condition naturally (see lemma 1). So \( L^s \) is invertible.

Based on the above theorem, we can compute the initial classification vector using Eq.(3.14), in which we only need to compute the inverse of an \( n^s \times n^s \) matrix, and usually \( n^s \) is much smaller than \( n \).

### 3.2.3 Solution Refining

Having achieved the initial classification vector from Eq.(3.14), we have to refine it level by level to get a classification vector on the initial graph \( G^0 = G \). As stated in section 3.2.1, we assume that the classification vector on graph \( G^l \) can be linearly interpolated from \( G^{l+1} \), i.e. \( f^l = P^{[l+1]} f^{l+1} \). Here \( P^{[l+1]} \) is an \( n^l \times n^{l+1} \) interpolation matrix subject to \( \sum_j P^{[l,0]}_{ij} = 1 \).

Based on the simple geometric intuition that the label of a point should be similar to the label of its neighbors (which is also consistent with the cluster assumption we introduced in section 1), we propose to compute \( P^{[l+1]}_{ij} \) by
\begin{equation}
(3.15) \quad P^{[l+1]}_{ij} = \begin{cases} 
\frac{w_{ij}}{\sum_{k \in C^l} w_{ik}} & i \notin C^l \\
1 & i = j \\
0 & x_i \in C^l, \ i \neq j
\end{cases}
\end{equation}

In the above equation, subscripts \( i,j,k \) are used to denote the index of the nodes in \( V^l \). We assume that node \( j \) has been selected as a \( C \)-node, and \( I(j) \) is the index of \( j \) in \( V^{l+1} \). It can be easily inferred that \( P^{[l+1]} \) has the following property.

**Lemma 3.1.** The interpolation matrix \( P^{[l+1]} \) has full rank.
Thus Lemma 4.1. The Eq. (3.14) will also cause a system of equations:

\[ [R : I_{l+1}] \text{ where } I_{l+1} \text{ is an } n^{l+1} \times n^{l+1} \text{ identity matrix. Thus } P^{[l,l+1]} \text{ has full rank.} \]

After all the interpolation matrices having been calculated using Eq. (3.8), we can use them to interpolate the initial classification vector level by level until we get a solution of the initial problem.

3.3 A Toy Example

To provide an intuitive illustration of the proposed method, we first give a toy example in Fig. 2. The leftmost figure of Fig. 2 shows the original problem, which is a two-circle pattern containing 2252 data points, with only two labeled. From left to right, Fig. 2 shows the coarsened data graphs from level 1 to level 3. On all these 4 graph levels, our method can correctly classify all the data points. On our Pentium IV 2.2G Hz computer, directly predicting the data labels on the original dataset (level 0) will take 25.7670 seconds, but it only takes 2.1330 seconds using our multilevel approach with the data graph coarsened to level 4, and both the methods can produce correct result.

4 Relationship with Multigrid Methods

Multigrid methods [17] are a set of methods that originally developed to solve boundary value problems posed on spatial domains. Such problems can be made discrete by choosing a set of grid points in the domain of the problem. The resulting discrete problem is a system of algebraic equations.

If we treat the data graph \( G \) as an Algebraic grid [17], then minimizing Eq.(2.3) will also result in a system of linear equations on these grid points (i.e. data points) since \( \partial J(f)/\partial f = 0 \) will cause

\[
(S + \gamma I)f = \gamma y.
\]

If we use \( S = S^c = D^{-1/2}(D-W)D^{-1/2} \) as in [14], then we can easily draw an interesting conclusion:

**Lemma 4.1.** The iterative framework proposed in [14] is just a Jacobi iteration [17] for minimizing

\[
J^c = f^T S f + \gamma \|f - y\|^2,
\]

which implies that we can apply the multigrid methods to accelerate Zhou’s consistency method [14].

**Proof.** See Appendix 2.

Furthermore, on a coarse level, the minimization of Eq. (3.14) will also cause a system of equations:

\[
(P^{[s,s-1]} \ldots P^{[1,0]}(S + \gamma I)P^{[0,1]} \ldots P^{[s-1,s]}) f^s = \gamma P^{[s,s-1]} \ldots P^{[1,0]} y
\]

which is just the coarse level Algebraic Multigrid (AMG) systems for Eq.(4.16) based on the Galerkin principal [17]. The main difference between the two methods is that: the AMG method approximates the solution error on a coarse grid (see [17] for details), while our method directly approximates the solution on a coarse grid due to the specific problem background and the cluster assumption, which is much faster. Moreover, if we define \( S \) as in Eq. (3.10), then we can derive the following theorem.

**Theorem 4.1.** Let \( P^{[0,s]} = P^{[0,1]} \ldots P^{[s-1,s]}, P^{[s,0]} = (P^{[s,s]}|S + \gamma I)^T, Q^0 = S + \gamma I, Q^s = P^{[s,0]} Q^0 P^{[0,s]} \). Define the \( Q^0 \) norm of a vector \( u \in \mathbb{R}^{n_o} \) to be

\[
\|u\|_{Q^0} = \sqrt{u^T Q^0 u}.
\]

and \( R(P^{[s,0]}) \) is the column space of \( P^{[s,0]} \). Clearly, \( f^s \in R(K) \). Let \( f^s = f^* - \gamma P^{[0,s]}(Q^s)^{-1} P^{[s,0]} y \) be the prediction error vector, then

\[
\|e\|^2_{Q^0} = \min_{v \in R(P^{[0,s]})} \|f^* - v\|^2_{Q^0}.
\]

**Proof.** See Appendix 3.

Theorem 3 implies our method can give a good approximation to the solution of the original problem.

5 Experiments

In this section, we give a set of experiments to show the effectiveness of our multilevel approach. In all of the experiments, the regularization parameter \( \gamma \) in Eq.(2.3) is set to 0.01, and the control parameter \( \delta \) is set to 0.2. All the experiments are performed on a Windows machine with 2.2 GHz Pentium IV processor and 512 MB main memory.

5.1 A Synthetic Example

Fig. 3(a) shows a synthetic dataset with three classes, each being a horizontal band containing 1050 data points with only 1 point labeled. In our method, we construct the neighborhood graph with the number of nearest neighbors \( k = 5 \). We apply our method on this dataset with the graph level \( l \) varying from 0 to 30. On each graph level our method can correctly classify all the data points.

Fig. 3(b) shows how the running time and graph size vary when we coarsen the graph level by level. Clearly, the computational time can be significantly improved in each step when \( l < 5 \), since the size of the graph can be reduced greatly. After \( l > 14 \), there are only three points remaining in the graph (which are the three labeled points), so further coarsening will not help to speed up the algorithm.
Figure 2: Graph coarsening procedure on a two-circle pattern. The number of nearest neighbors $k$ is set to 5, and the control parameter $\delta$ in Eq.(3.7) is set to 0.2.

Figure 3: Experimental results of our multilevel graph based SSL on 3 band dataset. (a) shows the original dataset; the upper figure of (b) is the running time vs. graph level plot of our algorithm on the dataset in (a); the lower figure of (b) shows sizes of the graphs (i.e. number of nodes) on different levels; (c) shows plot of the running times of our algorithm (ordinate) vs. dataset scales (abscissa).
We also test the running time of our algorithm with the scale of the dataset varying. The experimental result is shown in Fig.3(c), where we also adopted the 3 band dataset with its size being $630m$, $(m = 1, 2, \ldots, 9)$. This figure tells us that with the growing of the dataset scale, the computational time of our algorithm will increase much slower on a coarser graph.

5.2 Digits Recognition
In this case study, we will focus on the problem of classifying hand-written digits. The dataset we adopt is a subset of the USPS [18] handwritten 16x16 digits dataset. The images of digits 1, 2, 3 and 4 are used in these experiments as four classes, and there are 1269, 929, 824 and 852 examples in each class, with a total of 3874. The same dataset has been used in [14] as a benchmark dataset.

We used Nearest Neighbor classifier and one-vs-rest SVMs [11] as baselines. The width of the RBF kernel for SVM was set to 5 using a five fold cross validation. For comparison, we also provide the classification results achieved by Zhou et al.’s consistency method [14] and Zhu et al.’s Gaussian fields approach [16]. The affinity matrix in both methods were constructed by a Gaussian function whose variance is set by five fold cross validation. In our multilevel method, the number of nearest neighbors $k$ was set to 5 as in [12].

The recognition accuracies averaged over 50 independent trials are summarized in Fig.5(a), in which our method directly performs on the original data graph, i.e. graph $G^0$. We can see that without the tedious work of tuning parameters, our method can still produce comparable results with Zhou’s consistency method. Fig.5(b) provides the average recognition accuracies achieved by our method on different graph levels, which shows that the classification results may be affected more on coarser graphs when labeled set is too small.

The computational time vs. graph level plot is shown in Fig.4, from which we can see that learning on a coarser graph can save much computational time. Therefore, as long as the labeled data is not too few, our multilevel method can produce almost the same classification results in a much shorter time.

5.3 Text Classification
In this experiment, we addressed the task of text classification using 20-newsgroups dataset [19]. The topic rec containing autos, motorcycles, baseball and hockey was selected from the version 20news-18828. The articles were processed by the Rainbow software package with the following options: (1) passing all words through the Porter stemmer before counting them; (2) tossing out any token which is on the stoplist of the SMART system; (3) skipping any headers; (4) ignoring words that occur in 5 or fewer documents. No further preprocessing was done. Removing the empty documents, we obtained 3970 document vectors in a 8014-dimensional space. Finally the documents were normalized into TF-IDF representation.

We use the inner-product distance to find the $k$ nearest neighbors when constructing the neighborhood graph, i.e.

\[
(5.21) \quad d(x_i, x_j) = 1 - \frac{x_i^T x_j}{\|x_i\| \|x_j\|}
\]

where $x_i$ and $x_j$ are document vectors. And the value of $k$ is set to 10 manually. When using preprocessing, the threshold of confusion rate is chosen such that 2% of the data are thrown. For Zhou’s consistency and Zhu’s Gaussian fields methods, the affinity matrices were all computed by

\[
(5.22) \quad (W)_{ij} = \exp \left( -\frac{1}{2\sigma^2} \left( 1 - \frac{x_i^T x_j}{\|x_i\| \|x_j\|} \right) \right),
\]

The SVM and Nearest Neighbor classifiers were also served as the baseline algorithms. And all the hyperparameters in these methods were set by a five fold cross validation. Fig.6(a) shows the average classification accuracies those methods, from which we can clearly see the superiority of our method.

Fig.6(b) provides the average recognition accuracies achieved by our method on different graph levels, and the average computational time for implementing one prediction on the graph of different levels is shown in Fig. 7. Clearly, our multilevel scheme can effectively approximate the solution of the original problem with much shorter time as long as the size labeled data set is not too small.
Figure 5: Multilevel graph based SSL on USPS dataset. In both figures, the ordinate is the recognition accuracy averaged on 50 random trials, and the abscissa represents the number of points we randomly labeled, where we guarantee that there is at least one labeled point in each class. (a) classification results of different methods; (b) shows the classification results on different graph levels.

Figure 6: Multilevel graph based SSL on 20Newsgroup dataset. In both figures, the ordinate is the recognition accuracy averaged on 50 random trials, and the abscissa represents the number of points we randomly labeled, where we guarantee that there is at least one labeled point in each class. (a) classification results of different methods; (b) shows the classification results on different graph levels.
6 Conclusions
In this paper we propose a novel multilevel approach for graph based semi-supervised learning which can solve the SSL problem in a multilevel way. The feasibility of our method is analyzed theoretically in this paper, and the experiments show that our method can produce comparable results with traditional methods with much shorter time.

Acknowledgements
The authors would like to thank the constructive comments of the anonymous reviewers. This work is supported by project (60675009) of the National Natural Science Foundation of China.

Appendix 1: Proof of theorem 3.1
The smoothness of the classification function on the graph of level $s + 1$ is

$$S(f^{s+1}) = \sum_{k,l} w_{kl} (f_{k}^{s+1} - f_{l}^{s+1})^2$$

$$= \frac{1}{2} \sum_{k,l} \sum_{i,j} w_{ij} \left( p_{ji}^{[s,s+1]} - p_{il}^{[s,s+1]} \right) \cdot \left( p_{ij}^{[s,s+1]} - p_{lj}^{[s,s+1]} \right) \left( f_{k}^{s+1} - f_{l}^{s+1} \right)^2$$

$$= \frac{1}{2} \sum_{i,j} w_{ij} \sum_{k,l} \left( p_{i}^{[s,s+1]} p_{j}^{[s,s+1]} - p_{il}^{[s,s+1]} p_{jk}^{[s,s+1]} \right) \cdot \left( f_{k}^{s+1} - f_{l}^{s+1} \right)^2$$

Since $\sum_{l} p_{jl}^{[s,s+1]} = 1$, $\sum_{l} p_{il}^{[s,s+1]} = 1$, $\sum_{k} p_{jk}^{[s,s+1]} = 1$, $\sum_{k} p_{ik}^{[s,s+1]} = 1$, thus

$$\sum_{k,l} \left( p_{jl}^{[s,s+1]} - p_{il}^{[s,s+1]} \right) \left( p_{jk}^{[s,s+1]} - p_{il}^{[s,s+1]} \right) = 0$$

And it can be easily verified that

$$\sum_{k,l} \left( p_{jl}^{[s,s+1]} - p_{il}^{[s,s+1]} \right) p_{jk}^{[s,s+1]} = 0$$

Therefore $S(f^{s+1}) = S(f^s)$.

Appendix 2: Proof of lemma 4.1
First let’s recall that the iteration framework proposed in [14] is

$$f^{i+1} = \alpha \tilde{W} f^i + (1 - \alpha) y,$$

where $f^i$ is the predicted label vector at step $t$ with $f^0 = y$, $\tilde{W} = \tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2}$ is the weighted similarity matrix with $\tilde{W}_{ij} = 0$.

We can transform the objective in Eq.(4.17) as

$$J^c = f^T S f + \gamma \| f - y \|^2$$

$$= f^T (I - \tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2}) f + \gamma (f - y)^T (f - y)$$

Let

$$\frac{\partial J^c}{\partial f} = 0$$

we can get

$$f^{i+1} = (I - (1 - \gamma) \tilde{W})^{-1} f = \gamma y$$

Let $\alpha = 1 - \gamma$, we can rewritten the above equation as

$$(I - \alpha \tilde{W})^{-1} f = (1 - \alpha) y,$$

which is a system of linear equations. Using the Jacobi iteration [7], we can solve the above equation by updating

$$f^{i+1} = R_{f} f^i + \hat{D}^{-1} \hat{y},$$

iteratively, where $f^i$ is the predicted solution at step $t$, $\hat{y} = (1 - \alpha) y$, $R_{f} = \hat{D}^{-1} (L + U) = \alpha \tilde{W}$. Here $\hat{D} = I$.
is the diagonal of the matrix $A = I - W, -L$ and $-U$ are the strictly lower and upper triangle matrices of $A$. Therefore Eq.(6.26) becomes

$$f^{t+1} = \alpha Wf^t + (1-\alpha)y.$$  

If we set the initial point $f^0 = y$, then Eq.(6.27) is just the iteration equation of Zhou's consistency method. □

**Appendix 3: Proof of theorem 4.1**

According to Eq.(3.9), we can derive the cost function that we want to minimize in step $s$ as

$$J(f^s) = (f^s)^T P^{[s,0]} SP^{[s,0]} f^s + \gamma \| P^{[s,0]} f^s - y \|^2$$

Let $\partial J(f^s)/\partial f^s = 0$ we get the following linear equation system

$$(6.29) \quad P^{[s,0]} (S + \gamma I) P^{[s,0]} f^s = \gamma P^{[s,0]} y$$

Let $Q^s = P^{[s,0]} (S + \gamma I) P^{[s,0]} f^s$, then

$$(6.30) \quad f^s = \gamma (Q^s)^{-1} P^{[s,0]} y$$

Thus the predicted label vector at the initial graph is

$$(6.31) \quad f^0 = P^{[0,0]} f^s = \gamma P^{[0,0]} (Q^s)^{-1} P^{[0,0]} y$$

Let $Q^0 = S + \gamma I$, and $f^*$ be the exact solution to Eq.(4.16), then the prediction error vector is

$$(6.32) \quad e = f^* - P^{[0,0]} (Q^s)^{-1} P^{[0,0]} Q^0 f^*$$

Let $M = I - P^{[0,0]} (Q^s)^{-1} P^{[0,0]} Q^0$. Since

$$M^2 = I - 2P^{[0,0]} (Q^s)^{-1} P^{[0,0]} Q^0 + P^{[0,0]} (Q^s)^{-1} P^{[0,0]} Q^0 P^{[0,0]} (Q^s)^{-1} P^{[0,0]} Q^0 = I - P^{[0,0]} (Q^s)^{-1} P^{[0,0]} Q^0 = M,$$

the matrix $M$ is idempotent. Moreover, define the $Q^0$-inner product of two vectors $a$ and $b$ as

$$(6.33) \quad <a, b>_{Q^0} = <Q^0 a, b>,$$

where $<\cdot, \cdot>_{E}$ is the common Euclidean inner product. Since $Q^0$ and $Q^0M$ is symmetric, thus

$$<Ma, b>_{Q^0} = <Q^0Ma, b> = (Q^0Ma)^T b = a^T (Q^0M) b = a^T (Q^0Mb) = <Q^0a, Mb>_{Q^0},$$

i.e. $M$ is Hamiltonian with respect to the $Q^0$-inner product. Therefore, $M$ is an orthogonal projector. And for $\forall a \in \mathbb{R}^n, b \in \mathbb{R}^n$, we have

$$<Ma, P^{[0,0]} b>_{Q^0} = <Q^0Ma, P^{[0,0]} b> = a^T (Q^0 - Q^0P^{[0,0]} (Q^s)^{-1} P^{[0,0]} Q^0)P^{[0,0]} b = 0$$

i.e. the column space of $M$ ($\mathcal{R}(M)$) is orthogonal to the column space of $P^{[0,0]}$ ($\mathcal{R}(P^{[0,0]}$) w.r.t the $Q^0$ norm.

Therefore for $\forall u \in \mathcal{R}(M), v \in \mathcal{R}(P^{[0,0]}$), we have

$$(6.34) \quad \|u + v\|_{Q^0}^2 = \|u\|_{Q^0}^2 + \|v\|_{Q^0}^2$$

Here the $Q^0$-norm is defined as

$$\| \cdot \|_{Q^0} = \sqrt{\langle \cdot, \cdot \rangle_{Q^0}}$$

An observation here is that

$$I - M = P^{[0,0]} (Q^s)^{-1} P^{[0,0]} Q^0.$$ 

So $\mathcal{R}(I - M) = \mathcal{R}(P^{[0,0]}$), and

$$\|Mu + (I - M)u\|_{Q^0}^2 = \|Mu\|_{Q^0}^2 + \|(I - M)u\|_{Q^0}^2$$

Therefore

$$\min_{v \in \mathcal{R}(P^{[0,0]})} \|f^* - v\|_{Q^0}^2 = \min_{v \in \mathcal{R}(P^{[0,0]})} \|Mf^* + (I - M)f^* - v\|_{Q^0}^2 = \min_{v \in \mathcal{R}(P^{[0,0]})} \|Mf^* - v\|_{Q^0}^2 = \min_{v \in \mathcal{R}(P^{[0,0]})} \|Mf^*\|_{Q^0}^2 + \|v\|_{Q^0}^2 = \|Mf^*\|_{Q^0}^2 = \|e\|_{Q^0}^2,$$

which proves the theorem. □

**References**


