Lightweight Label Propagation for Large-Scale Network Data[*]

De-Ming Liang and **Yu-Feng Li**[†]
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, 210023, China
{liangdm, liyf}@lamda.nju.edu.cn,

Abstract

Label propagation spreads the soft labels from few labeled data to a large amount of unlabeled data according to the intrinsic graph structure. Nonetheless, most label propagation solutions work under relatively small-scale data and fail to cope with many real applications, such as social network analysis, where graphs usually have millions of nodes. In this paper, we propose a novel algorithm named SLP to deal with large-scale data. A lightweight iterative process derived from the well-known stochastic gradient descent strategy is used to reduce memory overhead and accelerate the solving process. We also give a theoretical analysis on the necessity of the warm-start technique for label propagation. Experiments show that our algorithm can handle million-scale graphs in few seconds while achieving highly competitive performance with existing algorithms.

1 Introduction

With the vigorous development of the Internet, many enterprises, such as Twitter, Facebook, YouTube etc., are accumulating massive data in daily operating. Information which can boost revenue are buried under raw data and need to dig out. Among them, the social network is one of worthiest data to mine and classification on a network would provide useful information on the nodes which represent users.

Label propagation algorithms [Zhu *et al.*, 2003; Joachims, 2003; Zhou *et al.*, 2004] work well in the setting with a graph and small amount of annotation, which aim to spread the soft labels from few annotated nodes to the whole graph according to the intrinsic structure. It's such a simple and effective method that it is widely used in practice. The key assumptions behind label propagation [Zhou *et al.*, 2004] are: (1) nearby nodes have the same labels; (2) nodes in the same structure tends to have the same label. One of the challenging issues of label propagation is the scalability. In general, it costs $O(kn^2)$ time to solve the linear system (k is the number of neighbors and n is the sample size). What's worse is that the graph matrix costs $O(n^2)$, which is a disaster when applied to large-scale graphs.

Various algorithms have been proposed to mitigate the scalability problem. Most of them mainly focus on graph approximation to improve the efficiency and can be divided into two sorts: low-rank approximation and sparse approximation.

The low-rank approximation is implemented in several ways. [Wang and Zhang, 2008; Zhang *et al.*, 2009; Liu *et al.*, 2010; Liu *et al.*, 2012; Lever *et al.*, 2012] select a subset of the nodes as the prototypes and use them in the following predicting procedure. Empirically, the number of prototypes m should be proportional to n, and they are usually selected from the clustering centers, which makes the prototypes expensive and inefficient. An alternative approach is based on the fast spectral decomposition of Laplacian matrix [Fergus *et al.*, 2009; Talwalkar *et al.*, 2013]. They try to find a representation of the structure in a low-dimension space formed by the eigenvectors of the Laplacian matrix with smallest eigenvalues and then train a classifier on this representation. The decomposition computation needs careful design to accelerate, and it's still inefficient when the graphs are highly sparse.

Graph sparse approximation mainly utilizes the minimum spanning tree (MST). These approaches first construct a minimum spanning tree on the given graph, and then solve the label propagation with a designed tree Laplacian solver [Herbster *et al.*, 2009; Cesa-Bianchi *et al.*, 2009; Vitale *et al.*, 2011; Zhang *et al.*, 2016]. These approaches seem great in complexity analysis, but they need to solve each connected component respectively, and the constructing process needs meticulous coding work.

In addition, Fujiwara and Irie [2014] propose a scalable algorithm different from those above. They compute the lower and upper bounding label scores and iteratively prune unnecessary score computations for acceleration. However, the selection of coefficient for bounding scores may be tricky. The iteration process either terminates in a single epoch or fails to stop, which may end up with poor performance.

In this paper, we try to ease the burden of large-scale label propagation in a way distinct from the approximation work mentioned above. We solve LP (label propagation) as a semi-supervised learning problem, which is equivalent to minimize a cost function from the graph and derive a

[*]This research was supported by the National Key R&D Program of China (SQ2018YFB100002) and the National Natural Science Foundation of China (61772262).

[†]Yu-Feng Li is the co-first author.

Table 1: Summary of Notation

Notation	Meaning
n	Number of instances
l	Number of labeled instances, $l \ll n$
$\boldsymbol{x}_i \in \mathbb{R}^d$	Feature vector of instance
$y_i \in \{+1, -1\}$	Label of instance
$\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$	Labeled instances
$\{\boldsymbol{x_i}\}_{i=l+1}^{n}$	Unlabeled instances
$\mathbf{y}_{\boldsymbol{l}} = [y_1, y_2, ..., y_l]$	Label vector for given labeled instances
$f : \mathbb{R}^d \to [-1, 1]$	Predictive function
$\mathbf{f}_{\boldsymbol{l}} = (f_1, f_2, ..., f_l)$	f's prediction on labeled instances
$\mathbf{f}_{\boldsymbol{u}} = (f_{l+1}, ..., f_n)$	f's prediction on unlabeled instances
$\mathbf{f} = [\mathbf{f}_{\boldsymbol{l}}, \mathbf{f}_{\boldsymbol{u}}]$	Predictive vector
diag$(\cdot)$	Diagonal matrix with diagonal vector $\cdot$
$\mathbf{1}$	Vector where all elements are set to 1
$\mathbf{e}_{\boldsymbol{i}}$	Vector where the i-th element is set to 1 and others are set to 0
$\mathbf{I}$	Identity matrix
$\mathbb{I}(\cdot)$	$\mathbb{I}(\cdot) = 1$ if $\cdot$ is true, otherwise $\mathbb{I}(\cdot) = 0$
$\mathbf{W} \in \mathbb{R}^{n \times n}$	Affinity matrix of instances
$\mathbf{W}_i$	the i-th row of $\mathbf{W}$
$\mathbf{D} \in \mathbb{R}^{n \times n}$	$\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} W_{ij}$
$\mathbf{L} \in \mathbb{R}^{n \times n}$	Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$
$\mathbf{A} \circ \mathbf{B}$	Hadamard product of matrix $\mathbf{A}$ and $\mathbf{B}$: $(\mathbf{A} \circ \mathbf{B})_{ij} = \mathbf{A}_{ij}\mathbf{B}_{ij}$

lightweight iterative process from the stochastic gradient descent (SGD) algorithm. SGD has been successfully applied to large-scale machine learning problems [Bottou, 2010; Gemulla *et al.*, 2011]. Our proposal is simple, fast and scalable, and we name the algorithm SLP (Stochastic Label Propagation). Experiments show that our algorithm can handle million-scale graphs in few seconds while achieving highly competitive performance with existing algorithms.

The paper is organized as follows. Section 2 introduces the setting of label propagation. Our proposed method is presented in Section 3, followed by the analyses in Section 4. We report the experimental results in Section 5 and then give conclusive remark finally.

2 Preliminaries

To propagate labels, a graph $G = (V, E)$ with affinity matrix $\mathbf{W}$ is constructed to reflect the similarity of instances. V is exactly the set of the feature vectors of nodes, and E is a set of edges between node pairs. The weights between any node pair (i, j) are stored in the affinity matrix $\mathbf{W}$. W_{ij} reflects the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The weight matrix can be defined as

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

where σ is the hyper-parameter [Zhu *et al.*, 2003].

Like other LP methods, we try to learn a real value function $f : \mathcal{V} \to [-1, 1]$ to make prediction [Zhu *et al.*, 2003]. For ease of operation, the labeled instances are placed from the beginning as the Table 1 indicates. The goal of LP is to find a labeling which is consistent with the initial labeling and the similarity of the data. The former is commonly considered in supervised learning paradigm, which can be measured by

$$\|\mathbf{f}_{\boldsymbol{l}} - \mathbf{y}_{\boldsymbol{l}}\|^2.$$

The latter one is called *smoothness assumption* [Chapelle *et al.*, 2006] and followed by the form

$$\sum_{i,j=1}^{n} W_{ij}(f_i - f_j)^2 = \mathbf{f}^\top(\mathbf{D} - \mathbf{W})\mathbf{f} = \mathbf{f}^\top\mathbf{L}\mathbf{f}.$$

As stated by Chapelle *et al.* [2006], label propagation can be cast into a common framework which minimizes a quadratic cost function as

$$\mathcal{C}(\mathbf{f}) = \mathbf{f}^\top\mathbf{L}\mathbf{f} + \mu\|\mathbf{f}_{\boldsymbol{l}} - \mathbf{y}_{\boldsymbol{l}}\|^2 + \epsilon\|\mathbf{f}\|^2. \tag{1}$$

The last term is added to prevent some degenerate situations, for instance, when the graph has a connected component with no labeled sample [Chapelle *et al.*, 2006].

Eq.(1) is similar to the regularization framework for the iterative algorithm proposed by [Zhou *et al.*, 2004]. In the extreme case when $\mu \to \infty, \epsilon = 0$, minimizing Eq.(1) is equivalent to minimize $\mathbf{f}^\top\mathbf{L}\mathbf{f}$ under the constraint $\mathbf{f}_{\boldsymbol{l}} = \mathbf{y}_{\boldsymbol{l}}$, which is the setting in [Zhu *et al.*, 2003].

The cost function is convex and so minimized when the derivative is set to 0, i.e.,

$$\mathbf{f}^* = (\mathbf{S} + \frac{1}{\mu}\mathbf{L} + \frac{\epsilon}{\mu}\mathbf{I})^{-1}\mathbf{S}\mathbf{y},$$

where $\mathbf{S}$ is a diagonal matrix with $S_{ii} = \mathbb{I}(1 \le i \le l)$. We can get the labels by simple matrix inversion. However, computing the inverse takes $O(n^3)$ time and $O(n^2)$ memory in general, which makes it infeasible on large-scale datasets.

3 Stochastic Label Propagation

In this section, we describe our simple and efficient algorithm SLP concretely. First, we derivate the updating routine for the common cost function. Then, the general framework is applied to the case in [Zhu *et al.*, 2003].

General Framework

In the following, we start from the cost function in Eq.(1) and give the stochastic iterative solution to the optimization problem,

$$\min_{\mathbf{f}} \ \mathcal{C}(\mathbf{f})$$

where

$$\begin{aligned}\mathcal{C}(\mathbf{f}) &= \mathbf{f}^\top\mathbf{L}\mathbf{f} + \mu\|\mathbf{f}_{\boldsymbol{l}} - \mathbf{y}_{\boldsymbol{l}}\|^2 + \epsilon\|\mathbf{f}\|^2 \\ &= \sum_{i,j}^{n} W_{ij}(f_i - f_j)^2 + \mu\sum_{i=1}^{l}(f_i - y_i)^2 + \epsilon\sum_{i=1}^{n} f_i^2\end{aligned}$$

Let

$$\mathcal{C}_i(\mathbf{f}) = \sum_{j=1}^{n} W_{ij}(f_i - f_j)^2 + \mathbb{I}_{[l]}(i) \cdot \mu(f_i - y_i)^2 + \epsilon f_i^2$$

where $\mathbb{I}_{[l]}(i) = \mathbb{I}(1 \le i \le l)$, $i \in \{1, 2, ..., n\}$ represents the index of the chosen instance. We have $\mathbb{E}(\nabla\mathcal{C}_i(\mathbf{f})) = \frac{1}{n}\nabla\mathcal{C}(\mathbf{f})$. Specifically, according to the definition, we have

$$\mathcal{C}(\mathbf{f}) = \sum_{i=1}^{n} \mathcal{C}_i(\mathbf{f})$$

which implies that

$$\nabla \mathcal{C}(\mathbf{f}) = \sum_{i=1}^{n} \nabla \mathcal{C}_i(\mathbf{f})$$

If we randomly select a node with index i, we have $\Pr[i] = \frac{1}{n}$. The expectation of $\nabla \mathcal{C}_i(\mathbf{f})$

$$\begin{aligned}\mathbb{E}[\nabla \mathcal{C}_i(\mathbf{f})] &= \sum_{i=1}^{n} \nabla \mathcal{C}_i(\mathbf{f}) \cdot \Pr[i] \\ &= \frac{1}{n} \sum_{i=1}^{n} \nabla \mathcal{C}_i(\mathbf{f}) \\ &= \frac{1}{n} \nabla \mathcal{C}(\mathbf{f})\end{aligned}$$

Therefore, $\nabla \mathcal{C}_i(\mathbf{f})$ is an unbiased estimator of $\frac{1}{n}\nabla \mathcal{C}(\mathbf{f})$ where $\frac{1}{n}$ is a constant given a graph. We can now adopt SGD strategy [Robbins and Monro, 1951] to solve $\mathbf{f_u}$ by following updating

$$\mathbf{f}^{(t+1)} = \mathbf{f}^{(t)} - \eta \nabla \mathcal{C}_i(\mathbf{f}),$$

where the gradient

$$\nabla \mathcal{C}_i(\mathbf{f}) = (\mathbf{f} - f_i \mathbf{1}) \circ \mathbf{W}_i^\top + \left(\mathbb{I}_{[l]}(i) \cdot \mu(f_i - y_i) + \epsilon f_i\right) \boldsymbol{e_i}.$$

However, the gradient takes $O(n)$ multiplication, which terribly slows down the iteration on a large graph. We observe that most of the elements in the vector of $\nabla \mathcal{C}_i(\mathbf{f})$ are zeros. In addition, the number of non-zero elements is determined by the degree $\deg(\mathrm{v})$ of the node v, and $\deg(\mathrm{v})$ is usually much smaller than u. In order to prevent unnecessary multiplication of zeros, we cache the neighbors' indexes for all nodes in the graph first. Then we use them to compute non-zero elements of $\nabla \mathcal{C}_i(\mathbf{f})$ and update $\mathbf{f}$. Trivially, the computation cost for each iteration is $O(\deg(\mathrm{v}))$.

Although SGD has made great success in large-scale machine learning, two more techniques are required for the effectiveness on large graphs. The first one is the warm start, which is attracting more and more interest in recent years. We manually select labeled instances and propagate their labels to neighbors before the iteration. The second one is to traverse all the shuffled nodes in each epoch as suggested by [Bottou, 2012]. This process ensures that every node is accessed in each epoch and so as to reduce variance of $\nabla \mathcal{C}_i(\mathbf{f})$.

The overall description of the framework is presented in Algorithm 1. It takes affinity matrix $\mathbf{W}$, label vector $\mathbf{y}_l$, step size η, epochs T and two coefficients of regularization term as input. After the initialization in line 1, it stores neighbors' indexes in $\mathcal{I}$. Then, the warm-start process is taken from line 3 to 7, where every node with an initial label is selected to update its neighbors' labels. Finally, it takes T epochs and performs n steps per epoch to update $\mathbf{f}$ from line 8 to 17. As suggested in [Bottou, 2012], the step size η is set to $O(\frac{1}{\sqrt{s}})$ where s is the total steps already taken. We call this framework SLP (Stochastic Label Propagation).

Algorithm 1 solves the regularization framework of label propagation with stochastic gradient and some techniques mentioned above. It gives a solution to large-scale regularization framework of label propagation and is easy to implement and light to run.

In the extreme case with $\mu \to \infty, \epsilon = 0$, minimize the quadratic cost function Eq.(1) is equivalent to minimize $\mathbf{f}^\top \mathbf{L} \mathbf{f}$ under the constraint of $\mathbf{f}_{\boldsymbol{l}} = \mathbf{y}_{\boldsymbol{l}}$, which is the setting in GRF [Zhu *et al.*, 2003]. More specifically, the problem is

$$\begin{aligned}\min_{\mathbf{f}} \;\; & \mathbf{f}^\top \mathbf{L} \mathbf{f} \\ s.t. \;\; & \mathbf{f}_{\boldsymbol{l}} = \mathbf{y}_{\boldsymbol{l}}\end{aligned}$$

We can derive corresponding algorithm for this problem from Algorithm 1 by removing the operations associated with μ, ϵ in line 14 as well as forcing $\boldsymbol{f_l} = \boldsymbol{y_l}$ in line 5 and line 12.

Algorithm 1 SLP Framework

```
Input: $\mathbf{W}$, $\mathbf{y}_l$, $\mu$, $\epsilon$, $\eta$, T
Output: $\mathbf{f}_u$
 1: Initialize $\mathbf{f}_u^{(1)}$ to $\mathbf{0}$, $\mathbf{f}_l^{(1)}$ to $\mathbf{y}_l$
 2: Find neighborhood indexes for node $i$ and store them in
    $\mathcal{I}(i), i = 1, 2, ..., n$
 3: for $i = 1, 2, ..., l$ do
 4:   for $ind$ in $\mathcal{I}(i)$ do
 5:     $f_{ind}^{(1)} \leftarrow f_{ind}^{(1)} - \eta(f_{ind}^{(1)} - f_i^{(1)})W_{i,ind}$
 6:   end for
 7: end for
 8: for $t = 1, ..., \mathrm{T}$ do
 9:   $r \leftarrow$ random index from $[1, n]$
10:   for $i = r, r+1, ..., n, 1, ..., r-1$ do
11:     for $ind$ in $\mathcal{I}(i)$ do
12:       $f_{ind}^{(t+1)} \leftarrow f_{ind}^{(t)} - \eta(f_{ind}^{(t)} - f_i^{(t)})W_{i,ind}$
13:     end for
14:     $f_i^{(t+1)} \leftarrow f_i^{(t)} - \eta(\mathbb{I}_{[l]}(i) \cdot \mu(f_i^{(t)} - y_i) + \epsilon f_i^{(t)})$
15:     update $\eta$
16:   end for
17: end for
```

4 Analyses

This section contains our analyses of SLP. In the first part, we give a convergent result and analyze the complexity of our algorithm. Then we discuss the necessity of warm start.

4.1 Convergence

Assume we have a dataset with n instances, the constructed graph $G = (V, E)$ has m edges, and the algorithm runs for T epochs. Adopting the proof from Chapter 14.3 of [Shalev-Shwartz and Ben-David, 2014], we have the following convergence result.

Theorem 1. *Let $B, \rho > 0$. Let $\mathbf{f}^* \in \arg\min_{\|\mathbf{f}\| \le B} \mathcal{C}(\mathbf{f})$. Assume that SLP run for T epochs with $\eta \le \sqrt{\frac{B^2}{\rho^2 T}}$, also assume that $\|\nabla \mathcal{C}_i(\mathbf{f})\| \le \rho$ with probability 1. Then,*

$$\mathbb{E}[\mathcal{C}(\mathbf{f})] - \mathcal{C}(\mathbf{f}^*) \le O(\frac{B\rho}{\sqrt{T}})$$

Therefore, for any $\epsilon > 0$, to achieve $\mathbb{E}[\mathcal{C}(\mathbf{f})] - \mathcal{C}(\mathbf{f}^*) \le \epsilon$, it suffices to run the algorithm for a number of epochs which satisfies $T \ge \frac{B^2\rho^2}{\epsilon^2}$.

In practice, the algorithm converges with a rather good solution in a few epochs (usually in 6 epochs). The inner two *for-loop* actually go through all edges in E and have a time complexity in $O(|E|)$, so the proposed algorithm can reach a good solution in $O(T|E|)$ time, which furthermore equivalent to $O(Tn)$ w.r.t sparse graphs. SLP needs to put the affinity matrix $\mathbf{W}$, the corresponding non-zero indexes $\mathcal{I}$ and predictive vector $\mathbf{f}$ into memory, so the space cost is $O(|E|)$.

4.2 Warm Start

In the following part, we try to explain why the warm-start process is required to perform in large-scale label propagation. We denote the length of the shortest walk between two nodes u and v as $dis(u,v)$. Given the initially labeled nodes, we can divide the V into several subsets $\hat{V}_l, V_1, V_2, ..., V_k$. $\hat{V}_l$ is the set of the initially labeled nodes, and $V_i = \{u|\min_{v\in V_l} dis(u,v) = i\}, i = 1,2,...,k$. The predictive label of a node is regarded as **valid** if it has received the label information from initially labeled nodes, and otherwise **invalid**.

Observation 1. *Based on the fact that label propagates through one edge per epoch, the predictions of the nodes in V_i will not be valid until the propagation of epoch i is done.*

We will show that the performance is affected by the density of graph and initially labeled ratio.

Theorem 2. *Let G be a graph with n nodes and the degree of all nodes in G is d. With l initial labeled nodes, after t-epoch propagation, at most $\min\{n, \frac{(d^{t+1}-1)l}{d-1}\}$ nodes will be valid.*

Proof. According to Observation 1, after t-epoch propagation, the predictions of the nodes in $\hat{V}_l, V_1, V_2, ..., V_t$ are valid, and we have

$$|\hat{V}_l \cup V_1 \cup ... \cup V_t| \le |\hat{V}_l| + |V_1| + ... + |V_t| \quad (2)$$
$$\le l + ld + ... + ld^t \quad (3)$$
$$= \frac{(d^{t+1}-1)l}{d-1}. \quad (4)$$

The two sides of inequality (2) are equal if and only if $\hat{V}_l, V_1, V_2, ..., V_t$ are mutual disjoint, and the two sides of inequality (3) are equal if and only if the graph G consists of d-ary trees and the roots of these trees are selected as initially labeled nodes.

Also notice that $|\hat{V}_l \cup V_1 \cup ... \cup V_t| \le n$, we have

$$|V_\ell \cup \hat{V}_l \cup ... \cup V_t| \le \min\{n, \frac{(d^{t+1}-1)l}{d-1}\},$$

and the theorem is proved. □

Lemma 1. *Let G be a graph with n nodes and the degree of all nodes in G is d. With l initial labeled nodes, the predictions on graph G will all be valid with t epochs, and $t \in \Omega(\log_d \frac{n(d-1)}{l})$.*

Proof. According to Theorem 2, after t-epoch propagation, at most $\min\{n, \frac{(d^{t+1}-1)l}{d-1}\}$ nodes will be valid. To make sure the predictions on the graph G are all valid, let $\frac{(d^{t+1}-1)l}{d-1} \ge n$, then we have

$$t \ge \log_d \left(\frac{n(d-1)}{l} + 1\right).$$

□

Theorem 2 and Lemma 1 show that the sparser the graph is, and the fewer the initial labels are, the more invalid predictions will be made in the first few epochs. The warm start process spreads initial labels to neighborhoods before main iteration (line 3 to 7 in Algorithm 1), which helps reduce invalid predictions when the algorithm terminates.

5 Experiments

We carry out three tasks to evaluate SLP. The first task is large-scale network analysis. In the second one, we compare SLP to several state-of-the-art algorithms on a categorization dataset. Finally, we show the influence of the density of graph on the performance of SLP.

Experimental Setup

The proposed approach is compared with a number of methods, including supervised method 10-NN as a baseline, ARG [Liu *et al.*, 2010], Eigen [Liu *et al.*, 2010] and TbTL [Zhang *et al.*, 2016]. Particularly, the ARG method uses a few anchor points which cover the entire point cloud to build a smaller graph with strong representative power. The final prediction is obtained by a simple linear combination of anchor points. The Eigen method approximates the graph Laplacian matrix by spectral decomposition, and in this way, the problem is solved in a reduced dimensional space. The TbTL method first generates a minimum spanning tree for given graph and then solve the problem with a designed Laplacian solver. The work presented in [Zhu *et al.*, 2003; Zhou *et al.*, 2004] fails to cope with the problem scale of our experiments, therefore, we do not include them in the experiments. For SLP, the extreme case mentioned at the end of Section 3 is used as the implementation and is evaluated in the following three tasks.

The codes of ARG, Eigen, TbTL are all shared by their authors. The parameters for comparison methods are chosen from the recommended ones suggested by the authors. We adopt KD-tree algorithm to construct approximate 10NN graphs when the dataset does not provide a graph. For ARG, anchor points are generated by 500-means clustering and the released LAE version is chosen to run. For Eigen, the number of eigenvectors is set to 160. For TbTL, the tree type is set to the minimum spanning tree, the number of trees is 16 as suggested by the authors and the regularization factor is selected from $0.01, 0.1, 1, 10$ by performing 5-fold cross-validation on the training set. We use default hyper-parameters for SLP. The step size η is set to $\frac{1}{10\sqrt{t}}$ where t is the next step to take and rounds T are set to 6. $0.1\%, 0.2\%, 0.4\%, 0.8\%, 1.6\%, 3.2\%$ of instances are randomly selected as labeled data. Results are averaged over 20 independent repetitions. The performance is measured in terms of both accuracy and AUC. We run these evaluations on a PC with 3.2GHz AMD Ryzen 1400 CPU and 16GB RAM.

Table 2: Label Propagation on Large-Scale Network Analysis

Graph	Labeled Ratio	Acc		Auc		Time(s)	
		TbTL	SLP	TbTL	SLP	TbTL	SLP
Wikipedia (1,791,489 nodes, 28,511,807 edges)	0.1%	.607 ± .009	.643 ± .008	.539 ± .006	.591 ± .006	33.1 ± 1.4	7.0 ± 0.4
	0.2%	.655 ± .003	.687 ± .004	.559 ± .004	.606 ± .004	37.2 ± 1.7	6.4 ± 0.1
	0.4%	.669 ± .002	.703 ± .002	.581 ± .003	.630 ± .005	45.6 ± 3.4	6.5 ± 0.2
	0.8%	.684 ± .001	.717 ± .002	.605 ± .002	.652 ± .004	60.2 ± 3.9	6.5 ± 0.1
	1.6%	.698 ± .001	.727 ± .001	.630 ± .001	.668 ± .003	90.6 ± 5.8	6.5 ± 0.1
	3.2%	.712 ± .001	.732 ± .001	.656 ± .001	.681 ± .003	146.0 ± 7.0	6.5 ± 0.1
LiveJournal (3,997,962 nodes, 34,681,189 edges)	0.1%	.979 ± .000	.950 ± .036	.515 ± .004	.619 ± .020	107.2 ± 5.4	10.3 ± 0.1
	0.2%	.979 ± .000	.960 ± .011	.530 ± .004	.646 ± .014	113.7 ± 4.4	10.2 ± 0.1
	0.4%	.979 ± .000	.964 ± .004	.549 ± .003	.670 ± .011	115.6 ± 4.2	10.4 ± 0.1
	0.8%	.980 ± .000	.966 ± .003	.575 ± .004	.696 ± .011	106.1 ± 8.6	10.4 ± 0.1
	1.6%	.981 ± .000	.969 ± .002	.614 ± .003	.732 ± .005	115.1 ± 6.3	10.5 ± 0.2
	3.2%	.982 ± .000	.973 ± .001	.657 ± .003	.769 ± .004	144.0 ± 8.7	10.5 ± 0.2
Orkut (3,072,441 nodes, 117,185,083 edges)	0.1%	.720 ± .010	.594 ± .045	.559 ± .007	.701 ± .012	256.5 ± 113.9	32.4 ± 1.9
	0.2%	.773 ± .003	.623 ± .013	.528 ± .004	.712 ± .010	222.5 ± 94.6	30.7 ± 0.6
	0.4%	.773 ± .001	.635 ± .008	.532 ± .002	.718 ± .007	248.9 ± 95.3	31.1 ± 0.5
	0.8%	.772 ± .001	.636 ± .007	.538 ± .002	.722 ± .003	232.9 ± 108.6	30.9 ± 0.6
	1.6%	.772 ± .001	.645 ± .005	.546 ± .001	.719 ± .002	247.9 ± 117.0	30.9 ± 0.5
	3.2%	.771 ± .001	.657 ± .004	.558 ± .001	.724 ± .002	205.7 ± 89.4	31.3 ± 0.4

Tasks on Large-Scale Network Analysis

We collect three large-scale network datasets and perform label propagation on them. These graphs [Yang and Leskovec, 2015; Yin *et al.*, 2017] include graphs from Wikipedia[1], Livejournal[2] and Orkut[3]. Specifically, Wikipedia is a web graph Wikipedia hyperlinks collected in September 2011 with $1,791,489$ nodes and $28,511,807$ edges. LiveJournal is a free on-line blogging community where users declare friendship each other and the graph contains $3,997,962$ nodes and $34,681,189$ edges. Orkut is a free on-line social network where users form friendship each other and the graph contains $3,072,441$ nodes and $117,185,083$ edges.

We can not access the profiles of nodes in these graphs and the labels of these nodes are generated by the ground-truth communities information provided by [Yang and Leskovec, 2015]. We adopt the one-vs-rest strategy by regarding the most frequent label as positive one. Because Anchor and Eigen methods take features of instances as input and fail to run in these tasks, we can only compare SLP with TbTL. We will evaluate Anchor and Eigen in the following part.

The performance is shown in Table 2. SLP achieves highly competitive performance with TbTL with much shorter time. Although TbTL seems to have higher accuracy, its poor AUC performance indicates that it benefits from the unbalance of the classes. We observe that most of TbTL's predictions are the label with the larger proportion in the training set. The superior of SLP over TbTL owes to the fact that SLP does not modify the given graph and maintain origin edges for label propagation. In addition, SLP is generally at least five times faster than TbTL.

[1]http://snap.stanford.edu/data/wiki-topcats.html

[2]http://snap.stanford.edu/data/com-LiveJournal.html

[3]http://snap.stanford.edu/data/com-Orkut.html

Tasks on Forest Covertype Categorization

The second task is collected from UCI[4] and the target is to predict forest cover type from cartographic variables only. On this dataset, more state-of-the-art methods can be conducted for comparison. The dataset has $581,012$ instances and we construct a 10NN graph based on the features. Following the procedure of network data, we transform the task into a binary task by treating the most frequent label as the positive label and the rest as the negative one.

Average accuracy, AUC and running time with standard deviation are shown in Table 3. On Covertype dataset, EIGEN, TbTL, and SLP are all inferior to 10NN and ARG with 0.1% and 0.2% labeled data. ARG may achieve better performance with more anchor points, but it's restricted by the memory cost. SLP performs better with more initial labeled data. Besides, given a graph, SLP spends much shorter time than competitors and achieve the highest AUC at most time.

Benchmark Tasks with Different Graphs

In order to evaluate how the density of the graph influences the performance of SLP, we further conduct experiments on several benchmark datasets collected from UCI[5] and MINIST[6] [LeCun *et al.*, 1998]. Six binary datasets, i.e., Adults, Ijcnn, Minist$3vs8$, Minist$4vs9$, Minist$7vs9$ and Devanagari are conducted. We construct 4NN, 6NN, 8NN and 10NN graphs for these datasets and compare the predictive accuracy of SLP with different graphs.

Results in Figure 1 show that our proposal can have a better performance on a denser graph. Commonly, 8NN and 10NN graphs give competitive results, while 4NN and 6NN graphs

[4]https://archive.ics.uci.edu/ml/datasets/Covertype

[5]https://archive.ics.uci.edu/ml/index.php

[6]http://yann.lecun.com/exdb/mnist/

Table 3: Comparison on Forest Covertype Categorization

	Method	0.1%	0.2%	0.4%	0.8%	1.6%	3.2%
Acc	10NN	$.684 \pm .007$	$\mathbf{.709 \pm .006}$	$.732 \pm .004$	$.758 \pm .003$	$.789 \pm .002$	$.822 \pm .001$
	ARG	$\mathbf{.688 \pm .008}$	$.703 \pm .005$	$.717 \pm .004$	$.728 \pm .002$	$.737 \pm .001$	$.741 \pm .001$
	Eigen	$.513 \pm .003$	$.530 \pm .017$	$.572 \pm .028$	$.629 \pm .004$	$.636 \pm .001$	$.646 \pm .004$
	TbTL	$.652 \pm .014$	$.698 \pm .007$	$.739 \pm .005$	$.777 \pm .003$	$.811 \pm .002$	$.840 \pm .001$
	SLP	$.633 \pm .011$	$.691 \pm .006$	$\mathbf{.742 \pm .004}$	$\mathbf{.787 \pm .003}$	$\mathbf{.825 \pm .002}$	$\mathbf{.857 \pm .001}$
Auc	10NN	$.687 \pm .007$	$.711 \pm .006$	$.734 \pm .004$	$.759 \pm .002$	$.790 \pm .002$	$.823 \pm .001$
	ARG	$\mathbf{.755 \pm .007}$	$\mathbf{.771 \pm .005}$	$.796 \pm .002$	$.812 \pm .002$	$.822 \pm .001$	$.828 \pm .001$
	Eigen	$.718 \pm .023$	$.724 \pm .023$	$.734 \pm .022$	$.734 \pm .009$	$.741 \pm .007$	$.752 \pm .003$
	TbTL	$.653 \pm .015$	$.695 \pm .008$	$.738 \pm .005$	$.778 \pm .003$	$.813 \pm .002$	$.845 \pm .001$
	SLP	$.731 \pm .011$	$.768 \pm .007$	$\mathbf{.801 \pm .004}$	$\mathbf{.837 \pm .003}$	$\mathbf{.868 \pm .002}$	$\mathbf{.899 \pm .001}$
Time(s)	10NN	457.2 ± 35.3	615.7 ± 70.4	913.2 ± 12.1	1544.0 ± 55.5	2767.7 ± 64.5	5270.0 ± 745.8
	ARG	786.1 ± 0.0	786.1 ± 0.0	786.1 ± 0.0	786.1 ± 0.0	786.1 ± 0.0	786.1 ± 0.0
	Eigen	8.3 ± 0.3	8.2 ± 0.2	9.1 ± 1.7	8.9 ± 1.5	8.2 ± 0.1	8.3 ± 0.3
	TbTL	20.1 ± 1.2	25.0 ± 0.8	31.1 ± 1.0	38.3 ± 1.1	46.4 ± 1.4	55.2 ± 1.2
	SLP	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.9 \pm 0.0}$	$\mathbf{0.9 \pm 0.0}$	$\mathbf{0.9 \pm 0.0}$	$\mathbf{0.9 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$

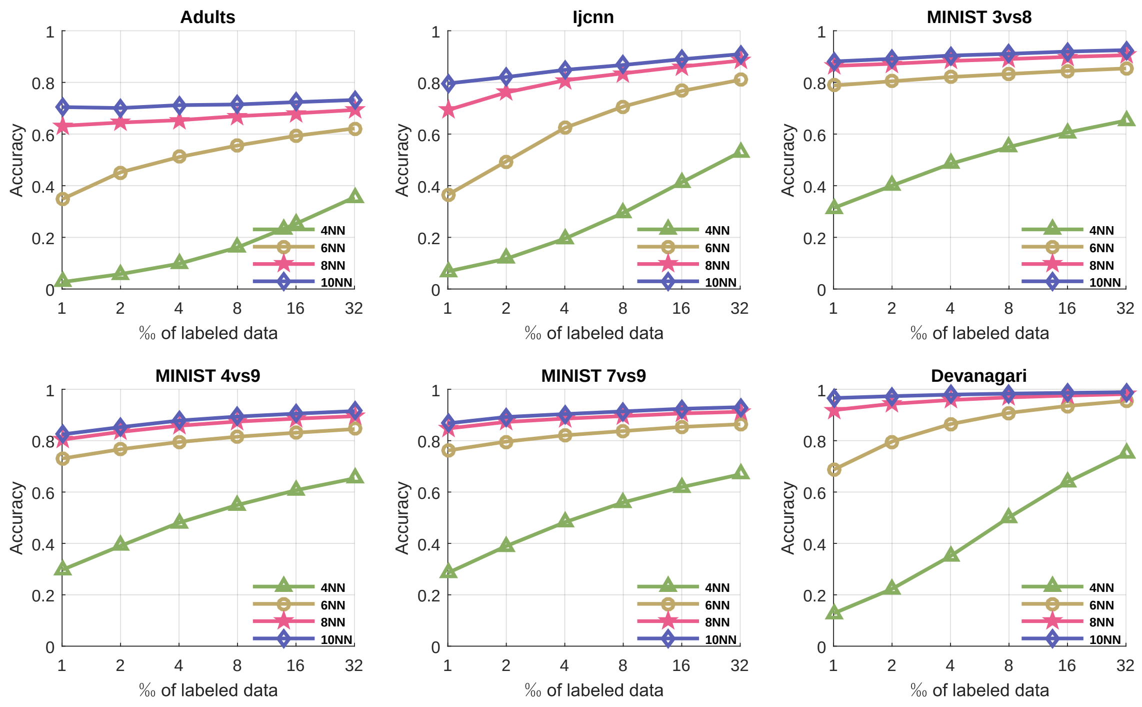

Figure 1: Performance of SLP with different graphs.

performs worse. This is consistent with Theorem 2. Particularly, the denser the graph is, the more predictions will be valid in the first few epochs.

6 Conclusion

In this paper, we propose a fast and scalable label propagation method named SLP. Its procedure is derived from stochastic gradient descent and it is simple to implement. The convergence as well as the necessity of warm start is analyzed. Experiments on million-scale graphs demonstrate that our method is nearly one order of magnitude faster than leading approaches and in addition achieves highly competitive or even the best performance. The contribution of this work is that we incorporate the stochastic method to label propagation and provides a different inspiration to handle large-scale label propagation tasks with clear practical advantages. There are many interesting future works. For example, our recent work showed that an inappropriate graph construction may deteriorate the performance of label propagation [Li *et al.*, 2016]. In future we will judge the quality of the graph for million-scale network analysis and derive robust label propagation by pruning inappropriate edges of graph.

References

[Bottou, 2010] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of International Conference on Computational Statistics*, pages 177–186. Springer, Paris, France, 2010.

[Bottou, 2012] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[Cesa-Bianchi *et al.*, 2009] Nicolo Cesa-Bianchi, Claudio Gentile, and Fabio Vitale. Fast and optimal prediction on a labeled tree. In *Annual Conference on Learning Theory*, pages 145–156, Montreal, Canada, 2009.

[Chapelle *et al.*, 2006] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-supervised learning*. MIT Press, 2006.

[Fergus *et al.*, 2009] Rob Fergus, Yair Weiss, and Antonio Torralba. Semi-supervised learning in gigantic image collections. In *Advances in Neural Information Processing Systems*, pages 522–530, Vancouver, Canada, 2009.

[Fujiwara and Irie, 2014] Yasuhiro Fujiwara and Go Irie. Efficient label propagation. In *Proceedings of the 31st International Conference on Machine Learning*, pages 784–792, Beijing, China, 2014.

[Gemulla *et al.*, 2011] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 69–77, San Diego, CA, 2011.

[Herbster *et al.*, 2009] Mark Herbster, Massimiliano Pontil, and Sergio R Galeano. Fast prediction on a tree. In *Advances in Neural Information Processing Systems*, pages 657–664, Vancouver, Canada, 2009.

[Joachims, 2003] Thorsten Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 290–297, Washington, DC, 2003.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Lever *et al.*, 2012] Guy Lever, Tom Diethe, and John Shawe-Taylor. Data dependent kernels in nearly-linear time. In *Artificial Intelligence and Statistics*, pages 685–693, La Palma, 2012.

[Li *et al.*, 2016] Yu-Feng Li, Shao-Bo Wang, and Zhi-Hua Zhou. Graph quality judgement: A large margin expedition. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 1725–1731, New York City, NY, 2016.

[Liu *et al.*, 2010] Wei Liu, Jun-Feng He, and Shih-Fu Chang. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on Machine Learning*, pages 679–686, Haifa, Israel, 2010.

[Liu *et al.*, 2012] Wei Liu, Jun Wang, and Shih-Fu Chang. Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, 100(9):2624–2638, 2012.

[Robbins and Monro, 1951] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[Shalev-Shwartz and Ben-David, 2014] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[Talwalkar *et al.*, 2013] Ameet Talwalkar, Sanjiv Kumar, Mehryar Mohri, and Henry Rowley. Large-scale svd and manifold learning. *Journal of Machine Learning Research*, 14(1):3129–3152, 2013.

[Vitale *et al.*, 2011] Fabio Vitale, Nicolo Cesa-Bianchi, Claudio Gentile, and Giovanni Zappella. See the tree through the lines: The shazoo algorithm. In *Advances in Neural Information Processing Systems*, pages 1584–1592, Granada, Spain, 2011.

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

[Yang and Leskovec, 2015] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[Yin *et al.*, 2017] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 555–564, Halifax, Canada, 2017.

[Zhang *et al.*, 2009] Kai Zhang, James T Kwok, and Bahram Parvin. Prototype vector machine for large scale semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1233–1240, Montreal, Canada, 2009.

[Zhang *et al.*, 2016] Yan-Ming Zhang, Xu-Yao Zhang, Xiao-Tong Yuan, and Cheng-Lin Liu. Large-scale graph-based semi-supervised learning via tree laplacian solver. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2344–2350, New Orleans, LA, 2016.

[Zhou *et al.*, 2004] Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 321–328, Vancouver, Canada, 2004.

[Zhu *et al.*, 2003] Xiao-Jin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912–919, Washington, DC, 2003.