Neural Process Lett (2015) 42:187–197
DOI 10.1007/s11063-014-9351-z

Scalable Semi-Supervised Classification via Neumann Series

Chen Gong · Keren Fu · Lei Zhou · Jie Yang · Xiangjian He

Published online: 16 April 2014

Abstract Traditional graph-based semi-supervised learning (GBSSL) algorithms usually scale badly due to the expensive computational burden. The main bottleneck is that they need to compute the inversion of a huge matrix. In order to alleviate this problem, this paper proposes Neumann series approximation (NSA) to explicitly approximate the inversion process required by conventional GBSSL methodologies, which makes them computationally tractable for relatively large datasets. It is proved that the deviation between the approximation and direct inversion is bounded. Using real-world datasets related to handwritten digit recognition, speech recognition and text classification, the experimental results reveal that NSA accelerates the speed significantly without decreasing too much precision. We also empirically show that NSA outperforms other scalable approaches such as Nyström method, Takahashi equation, Lanczos process based SVD and AnchorGraph regularization, in terms of both efficiency and accuracy.

Keywords Semi-supervised learning · Scalability · Neumann series · Error bound

1 Introduction

Semi-supervised learning (SSL) is suitable for the situations where labeled examples are scarce while unlabeled examples are extremely abundant. Graph-based SSL (GBSSL) has attracted intensive attention in recent years due to their promising performance and valid

C. Gong · K. Fu · L. Zhou · J. Yang (✉)
Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai, China
e-mail: jieyang@sjtu.edu.cn

C. Gong
e-mail: goodgongchen@sjtu.edu.cn

X. He
Department of Computer Science, University of Technology Sydney, Ultimo, NSW, Australia

theoretical basis. Various GBSSL algorithms such as Gaussian field and harmonic functions (GFHF) [22], local and global consistency (LGC) [21], manifold regularization (MR) [1], linear neighborhood propagation (LNP) [19], enhanced spectral kernel [12] and semi-supervised logistic discrimination [8], have been developed in recent years, and representatives of them are comprehensively reviewed in [23].

One drawback of these methods is that their scalability is rather poor. When a large graph is built, they have to calculate the inversion of a very huge matrix. This process is extremely time-consuming, and thus not suitable for large collections. In order to improve the scalability of SSL algorithms, some accelerating methods have been developed and they are grouped into three categories:

1. **Hyper-graph establishment:** Algorithms belonging to this category aim to reduce the size of graph by choosing a set of representative nodes. Zhu *et al.* [24] constructed the "backbone graph" by combining the generative mixture models and discriminative regularization. Delalleau *et al.* [3] approximated the training process by using only a subset of numerous examples. Liu *et al.* [10] located the critical anchor points with K-means and designed a regularization algorithm on this "AnchorGraph" (AGR).
2. **Sparse representation:** A GBSSL algorithm can be scalable if its solution is sparsely represented. Garcke *et al.* [6] proposed discretization technique by a sparse grid method, Tsang and Kwok [17] developed SLapCVM by using a sparsified manifold regularizer, and Sinha and Belkin [13] adopted sparse eigenfunction bases when the cluster assumption holds.
3. **Inversion approximation:** Some researchers attempt to design fast numerical techniques for inverting a large sparse matrix. Nyström low-rank approximation was adopted in many works, such as those in [5,16,18,20]. Larsen [9] computed the singular value decomposition (SVD) by Lanczos bidiagonalization algorithm with partial reorthogonalization, which could be used to approximating the inversion of large matrix by combining with the Woodbury matrix identity. Campbell and Davis [2] utilized the LDU factorization, and followed by Takahashi equation, to approximate the large sparse matrix. Moreover, Fergus *et al.* [4] adopted eigenvectors corresponding to small eigenvalues of graph Laplacian to represent the solution, which improved the inversion speed significantly.

The algorithms belonging to the first category sometimes cannot achieve satisfying performance because they discard considerable data information in the graph. The methods of the second category often need to solve a complex optimization problem, which will definitely slow down the computational speed. Therefore, the algorithm in this paper is designed by following the idea of last category and it is named as "Neumann series approximation" (NSA). We notice that the spectral radius of iteration matrix in a class of GBSSL algorithms, *e.g.* GFHF, LGC, LNP, is bounded by 1, so it is possible to use Neumann series [7] to approximate the inversion of the associated large sparse matrices. The error bound of this approximation is derived so that NSA can be theoretically validated. We demonstrate the strength of NSA by applying it to traditional GFHF, LGC and LNP algorithms. Massive experiments including digit recognition, speech recognition and text classification reveal that compared with [9,10,18] and naïve implementations of original GBSSL algorithms, NSA obtains satisfying classification results meanwhile needing very little computational time.

Table 1 Summarization of different SSL algorithms on label propagation

Algorithms	Propagating expressions
GFHF [22]	$\mathbf{f} = (\mathbf{I} - \alpha\tilde{\mathbf{W}})^{-1}\mathbf{y},\ \tilde{\mathbf{W}} = \mathbf{D}^{-1}\mathbf{W}$
LGC [21]	$\mathbf{f} = (\mathbf{I} - \alpha\mathbf{S})^{-1}\mathbf{y},\ \mathbf{S} = \mathbf{D}^{-1/2}\,\mathbf{W}\mathbf{D}^{-1/2}$
LNP [19]	$\mathbf{f} = (\mathbf{I} - \alpha\mathbf{W}_{LNP})^{-1}\mathbf{y}$

2 Semi-Supervised Algorithms

Given a set of labeled examples $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and a set of unlabeled examples $\mathcal{U} = \{(\mathbf{x}_i)\}_{i=l+1}^{l+u}$, typically with $l \ll u$, where $\mathbf{x}_i$ $(1 \le i \le n,\ n = l + u)$ are d-dimensional examples sampled from an unknown marginal distribution P_X, and y_i $(1 \le i \le l)$ are labels taking values from a binary label set $\{-1, 1\}$. A transductive SSL algorithm aims to infer the labels of u unlabeled examples $\{y_{l+1}, y_{l+2}, \dots, y_{l+u}\}$ based on the union of $\mathcal{L}$ and $\mathcal{U}$.

All examples $\{(\mathbf{x}_i)\}_{i=1}^{l+u}$ are usually represented by a graph $\mathcal{G}$ with the adjacency matrix denoted as $(\mathbf{W})_{ij} = \omega_{ij}$, in which $\omega_{ij} = \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2/2\sigma^2\right)$ is the RBF kernel. The degrees of n nodes form a diagonal matrix $(\mathbf{D})_{ii} = \sum_{j=1}^{n}\omega_{ij}$, and $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian of $\mathcal{G}$. Then, the label information is propagated from labeled examples to unlabeled examples on $\mathcal{G}$. Table 1 summarizes the expressions of some representative GBSSL algorithms, *i.e.*, GFHF, LGC and LNP, for label propagation. In Table 1, $\mathbf{f} = (f_1, f_2, \dots, f_n)^T$ records the final classification results and $\mathbf{y} = (y_1, \dots, y_l, 0, \dots, 0)^T$ is an n-dimensional label vector with $y_i = 1$ for positive examples and -1 for negative ones. α is a parameter usually set to 0.99 [19,21,22].

3 Neumann Series Approximation

From Table 1, we see that the propagating expressions in different algorithms can be cast into a unified expression:

$$\mathbf{f} = (\mathbf{I} - \alpha\mathbf{P})^{-1}\mathbf{y}, \tag{1}$$

where $\mathbf{P}_{n\times n} = \tilde{\mathbf{W}}, \mathbf{S}, \mathbf{W}_{LNP}$ for GFHF, LGC and LNP, respectively. However, $\mathbf{P}$ is a huge symmetrical matrix and inverting $\mathbf{I} - \alpha\mathbf{P}$ directly requires $O(n^3)$ complexity, so specific methods are required to lower the computational burden.

Lemma 1 ([14]) *If a matrix* $\mathbf{A}$ *has the property that* $\lim_{i\to\infty}(\mathbf{I} - \mathbf{A})^i = 0$, *then* $\mathbf{A}$ *is nonsingular and its inversion can be expressed by a Neumann series:*

$$\mathbf{A}^{-1} = \sum_{i=0}^{\infty}(\mathbf{I} - \mathbf{A})^i. \tag{2}$$

Theorem 2 *The matrix* $\mathbf{I} - \alpha\mathbf{P}$ *in* (1) *satisfies Lemma 1 and its inversion can be represented by*

$$(\mathbf{I} - \alpha\mathbf{P})^{-1} = \sum_{i=0}^{\infty}(\alpha\mathbf{P})^i. \tag{3}$$

Proof According to Lemma 1, we obtain $\lim_{i\to\infty}[\mathbf{I}-(\mathbf{I}-\alpha\mathbf{P})]^i=\lim_{i\to\infty}(\alpha\mathbf{P})^i$. Because $\alpha\in(0,1)$ and the spectral radius of $\mathbf{P}$, or $\rho(\mathbf{P})\leq 1$ [19,21,22], so from the *theorem of Perron-Frobenius* [7] we know that $\lim_{i\to\infty}(\alpha\mathbf{P})^i=0$. Therefore, (3) holds and thus we complete the proof. □

However, computing the summation of $(\alpha\mathbf{P})^i$ from $i=0$ to infinity is intractable, so we simply preserve the first t terms of Neumann series as the approximation of $(\mathbf{I}-\alpha\mathbf{P})^{-1}$, which arrives at:

$$(\mathbf{I}-\alpha\mathbf{P})^{-1}\approx\mathbf{I}+(\alpha\mathbf{P})^1+\cdots+(\alpha\mathbf{P})^t=\sum_{i=0}^{t}(\alpha\mathbf{P})^t. \tag{4}$$

Note that $\mathbf{P}$ is very sparse when $\mathcal{G}$ is a k-NN graph [23], so (4) avoids the inversion operation by calculating the power of sparse matrix, which is obviously very efficient. Moreover, the trade-off between efficiency and accuracy is controlled by the parameter t. Larger t usually leads to better approximation performance while the computational time will become longer. The details of choosing a proper t will be explained in Sect. 5.1.

4 Error Bound

In this section, we attempt to find the deviation bound of $(\mathbf{I}-\alpha\mathbf{P})^{-1}$ and its approximation.

Theorem 3 *The approximation error of NSA is bounded and the following inequality holds:*

$$\left\|(\mathbf{I}-\alpha\mathbf{P})^{-1}-\sum_{i=0}^{t}(\alpha\mathbf{P})^i\right\|_F\leq\frac{\alpha^{t+1}\sqrt{n}}{1-\alpha}. \tag{5}$$

Proof Because $\mathbf{P}\in\mathbb{R}^{n\times n}$ is symmetrical, it can be decomposed as $\mathbf{P}=\mathbf{U}\Lambda\mathbf{U}^T$ in which $\mathbf{U}$ is an orthogonal matrix and $\mathbf{\Lambda}=diag(\lambda_1,\lambda_2,\ldots,\lambda_n)$ is a diagonal matrix containing $\mathbf{P}$'s n eigenvalues. Then, according to Woodbury matrix identity,

$$(\mathbf{I}-\alpha\mathbf{P})^{-1}=(\mathbf{I}-\alpha\mathbf{U}\Lambda\mathbf{U}^T)^{-1}=\mathbf{I}+\alpha\mathbf{U}(\mathbf{\Lambda}^{-1}-\alpha\mathbf{I})^{-1}\mathbf{U}^T. \tag{6}$$

Therefore, the difference between $(\mathbf{I}-\alpha\mathbf{P})^{-1}$ and its approximation is

$$\begin{aligned}
\left\|(\mathbf{I}-\alpha\mathbf{P})^{-1}-\sum_{i=0}^{t}(\alpha\mathbf{P})^i\right\|_F &= \left\|(\mathbf{I}-\alpha\mathbf{U}\Lambda\mathbf{U}^T)^{-1}-\sum_{i=0}^{t}(\alpha\mathbf{U}\Lambda\mathbf{U}^T)^i\right\|_F\\
&= \left\|\mathbf{U}diag\left(\frac{1}{1-\alpha\lambda_1},\ldots,\frac{1}{1-\alpha\lambda_n}\right)\mathbf{U}^T-\mathbf{U}diag\left(\frac{1-(\alpha\lambda_1)^{t+1}}{1-\alpha\lambda_1},\ldots,\frac{1-(\alpha\lambda_n)^{t+1}}{1-\alpha\lambda_n}\right)\mathbf{U}^T\right\|_F\\
&= \left\|\mathbf{U}diag\left(\frac{(\alpha\lambda_1)^{t+1}}{1-\alpha\lambda_1},\ldots,\frac{(\alpha\lambda_n)^{t+1}}{1-\alpha\lambda_n}\right)\mathbf{U}^T\right\|_F\\
&= \sqrt{tr\left[\mathbf{U}diag\left(\frac{(\alpha\lambda_1)^{2t+2}}{(1-\alpha\lambda_1)^2},\ldots,\frac{(\alpha\lambda_n)^{2t+2}}{(1-\alpha\lambda_n)^2}\right)\mathbf{U}^T\right]}\\
&= \sqrt{\sum_{i=1}^{n}\frac{(\alpha\lambda_i)^{2t+2}}{(1-\alpha\lambda_i)^2}}.
\end{aligned} \tag{7}$$

Note that $\mathbf{P}$ is a stochastic matrix, so $\{\lambda_i\}_{i=1}^n \in [-1, 1]$ [19,21,22], therefore (7) is maximized when all the eigenvalues of $\mathbf{P}$ equal to 1 and

$$\left\| (\mathbf{I} - \alpha\mathbf{P})^{-1} - \sum_{i=0}^{t} (\alpha\mathbf{P})^i \right\|_F = \sqrt{\sum_{i=1}^{n} \frac{(\alpha\lambda_i)^{2t+2}}{(1-\alpha\lambda_i)^2}} \leq \frac{\alpha^{t+1}\sqrt{n}}{1-\alpha}. \tag{8}$$

Therefore, Theorem 3 is proved. Equation (8) also illustrates that the error bound will decrease with the increase of t (because $\alpha \in (0, 1)$), which is consistent with our general understanding. □

5 Experimental Results

In this section, we firstly give some empirical insight into the choice of parameter t, then NSA will be evaluated on some typical datasets related to handwritten digit recognition, speech recognition and text categorization. The traditional GBSSL algorithms like Gaussian field and harmonic functions (GFHF) [22], Local and Global Consistency (LGC) [21], linear neighborhood propagation (LNP) [19] and other scalable methods including Nyström approximation (Nyström) [18], Takahashi equation (TE, [2]), Lanczos process based SVD (SVD)[1] [9], AnchorGraph regularization (AGR) [10], serve as the baselines for comparison. Among them, Nyström and SVD approximate the matrix $\mathbf{P}$, and then employ the Woodbury matrix identity to further compute the inversion of $\mathbf{I} - \alpha\mathbf{P}$, while the remaining methods calculate $(\mathbf{I} - \alpha\mathbf{P})^{-1}$ directly. k-NN graph is adopted in the experiments below, based on which we observe the classification accuracies and computational time of different algorithms with the change of labeled set $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^l$. All algorithms are conducted on a working station with 2.40 GHz Intel Xeon CPU.

5.1 Choice of t

In our algorithm, t is the only parameter to be tuned, which harnesses the approximation performance and computational complexity. In order to choose a proper t, we observe the performance of GFHF, LGC, LNP and their combinations with NSA when $t = 1, 2, 3$. *Breast cancer* dataset[2] that contains 683 valid examples is adopted for illustration. 15-NN graph is constructed for all algorithms, and the width of RBF kernel σ is set to 1 for GFHF and LGC. The classification accuracy and time cost w.r.t different l are especially evaluated. For each l, each of the algorithms is implemented 10 times independently with randomly selected labeled examples. The final accuracies and time costs are calculated as the mean value of the outputs of these implementations. This measure is intended to eliminate the influence of the locations of initially labeled examples on the final output. Note that at least one labeled example is guaranteed in each class when the labeled sets are generated.

In the first row of Fig. 1, the accuracies of GFHF, LGC and LNP with error bars are plotted, and the average time costs for each l are presented in the second row. We observe that by applying NSA to the traditional GFHF, LGC and LNP, the time consuming drops significantly. However, the accuracy decreases not so much, especially when $t = 2, 3$. Generally speaking, NSA ($t = 2$) and NSA ($t = 3$) obtain similar encouraging classification accuracies. However, NSA ($t = 3$) takes much more time than NSA ($t = 2$) without significantly improving the

[1] Lanczos process based SVD is implemented by using PROPACK, which is available at http://soi.stanford.edu/~rmunk/PROPACK/.

[2] Available at http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29.

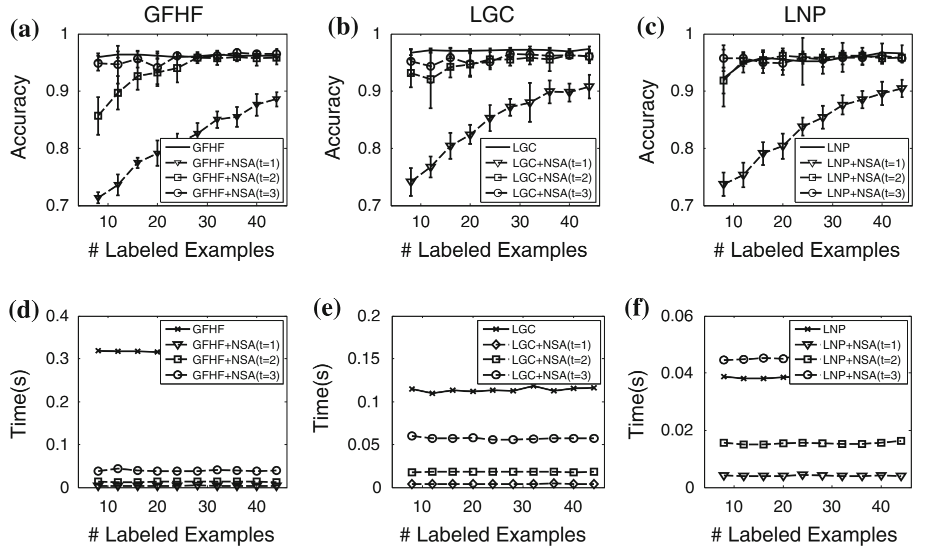

Fig. 1 Performances of NSA with different t. The three columns are GFHF, LGC and LNP respectively. The first row shows the accuracies of GFHF, LGC, LNP and their approximations for $t = 1, 2, 3$. The second row illustrates the computational time

classification performance, so t is set to 2 in our NSA algorithm. Note that when the labeled examples are extremely scarce, NSA with $t = 2$ may not achieve the comparable accuracy with standard GBSSL algorithms (*e.g.* GFHF and LNP in Fig. 1a, c). However, we think the performance of NSA is still acceptable. Alternatively, the users are suggested to label more examples before implementing NSA.

5.2 Handwritten Digit Recognition

Handwritten digit recognition is an important branch of optical character recognition (OCR). We adopt the *optical recognition of handwritten digits* dataset[3] to compare the proposed NSA with other baselines. This dataset contains 5,620 digital images corresponding to $0 \sim 9$, and the resolution of each image is 8×8 (see Fig. 2a). Therefore, the pixel-wise feature is described by a 64-dimensional vector with elements representing the gray levels.

All algorithms are conducted on a k-NN graph with $k = 15$, and the reported results are averaged over 10 independent runs with randomly selected examples for a given l. The standard deviations of accuracies are also recorded to see whether NSA is sensitive to the choice of initially labeled examples. For approximating $\mathbf{P}$, we retain the eigenvectors associated with the $p = 30$ dominant eigenvalues of $\mathbf{P}$ in SVD. In Nyström, we sample $c = 800$ columns uniformly and compute an approximated eigendecomposition of a rank-r ($r = 30$) matrix. In AGR, we set the number of anchor points to $m = 400$ and use Local Anchor Embedding (LAE) [10] to solve the associated optimization problem.

Figure 2b demonstrates that GFHF+NSA, LGC+NSA and LNP+NSA achieve higher accuracies than SVD, AGR, TE and Nyström when l changes from small to large. Besides, no matter how we tune the parameters c and r, we observe that the performance of Nyström approximation for large sparse matrix is far from satisfactory. This is due to the fact that

[3] Available at http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits.

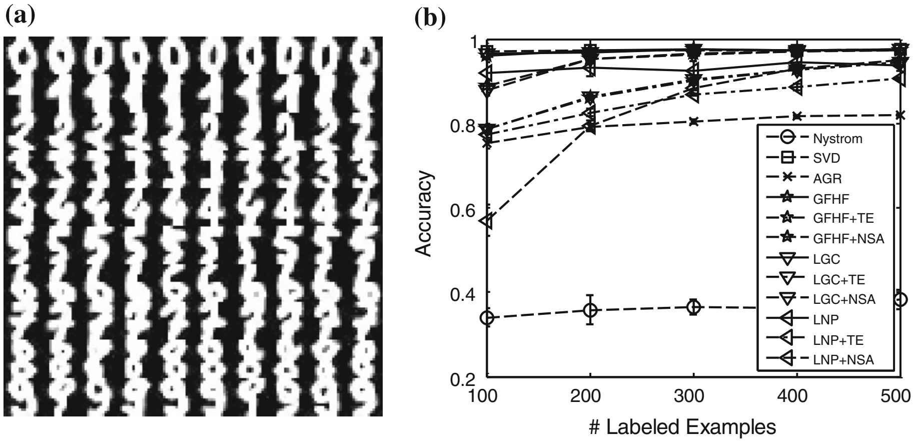

Fig. 2 Experiment on digit recognition. **a** Shows some typical examples in the dataset. **b** Plots the classification accuracies of various methods

Table 2 Time costs of different algorithms on digit recognition

Algorithms	Times (s)				
	$l = 100$	$l = 200$	$l = 300$	$l = 400$	$l = 500$
GFHF	72.1	72.0	73.2	73.8	73.3
LGC	10.8	10.8	11.1	10.8	10.7
LNP	13.0	13.6	12.8	13.0	12.8
GFHF + TE	2.1	2.1	2.2	2.2	2.2
LGC + TE	2.1	2.1	2.2	2.1	2.1
LNP + TE	3.3	3.3	3.3	3.3	3.3
GFHF + NSA	0.5	0.5	0.5	0.5	0.5
LGC + NSA	0.5	0.5	0.5	0.5	0.5
LNP + NSA	0.4	0.4	0.4	0.4	0.4
Nyström	3.7	4.1	4.0	4.0	3.6
SVD	0.9	0.9	0.9	0.9	0.9
AGR	0.2	0.2	0.2	0.2	0.2

the Nyström approximation only performs well when the approximated matrix is positive semidefinite. However, the matrix **P** to be approximated here is usually not positive semi-definite, which leads to the lower accuracy of Nyström approximation when compared with other algorithms. Therefore, here we set c and r to relatively small numbers by balancing both accuracy and efficiency. It is also observed that compared with the original implementations of GFHF, LGC and LNP, the accuracies of GFHF+NSA, LGC+NSA and LNP+NSA decrease very little. It is even more significant that the computational time is reduced greatly as Table 2 shows, which demonstrates the effectiveness and efficiency of NSA empirically. Besides, we observe that AGR is slightly faster than NSA. Nevertheless its accuracy is far lower than NSA. In addition, the error bars in Fig. 2b indicate that the accuracy of NSA is insensitive to the locations of initially labeled examples. Table 2 also reveals that the running time of all the methods is essentially irrelevant to the increase of l.

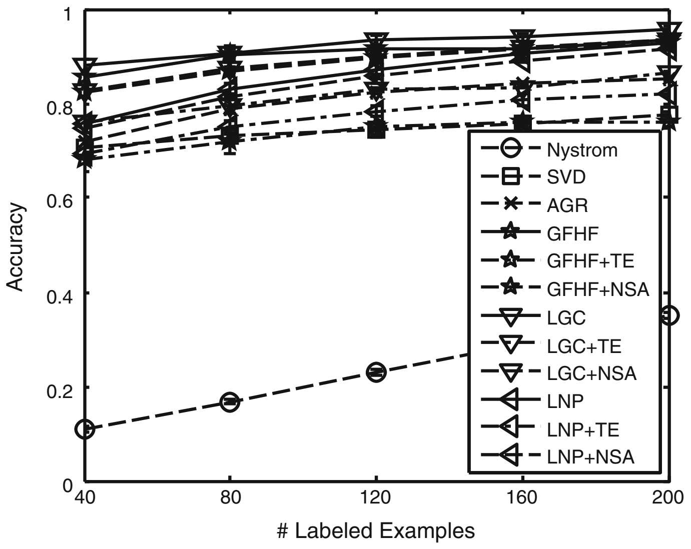

Fig. 3 Classification accuracies of various methods on *Isolet* dataset

Table 3 Time costs of different algorithms on *Isolet* dataset

Algorithms	Times(s)				
	$l = 40$	$l = 80$	$l = 120$	$l = 160$	$l = 200$
GFHF	195.7	195.6	196.2	197.2	197.5
LGC	34.6	41.0	41.6	32.8	31.6
LNP	35.7	35.5	35.7	36.2	35.5
GFHF + TE	45.6	44.3	45.3	46.9	47.6
LGC + TE	18.9	19.0	19.2	19.1	19.0
LNP + TE	34.0	32.9	31.8	31.8	31.6
GFHF + NSA	1.1	1.1	1.1	1.1	1.1
LGC + NSA	1.0	1.1	1.1	1.1	1.1
LNP + NSA	1.4	1.4	1.4	1.4	1.4
Nyström	7.5	6.8	7.2	7.9	7.1
SVD	1.9	1.9	1.9	1.9	1.9
AGR	0.9	1.4	1.0	1.0	0.9

5.3 Speech Recognition

In this experiment, we address a speech recognition task using the *Isolet* dataset[4]. In this dataset, 150 subjects are required to pronounce each letter in the alphabet twice. Excluding 3 missing examples, we have total $150 \times 2 \times 26 - 3 = 7797$ examples. Similar to the work in Sect. 5.2, we compare the accuracies and time costs of all algorithms for different values of l. We set $p = 50$ in SVD, $c = 150$, $r = 30$ in Nyström and $m = 700$ in AGR. The accuracies and computational time of NSA with SVD, AGR and Nyström are presented in Fig. 3 and Table 3, respectively.

[4] Available at: http://archive.ics.uci.edu/ml/datasets/ISOLET.

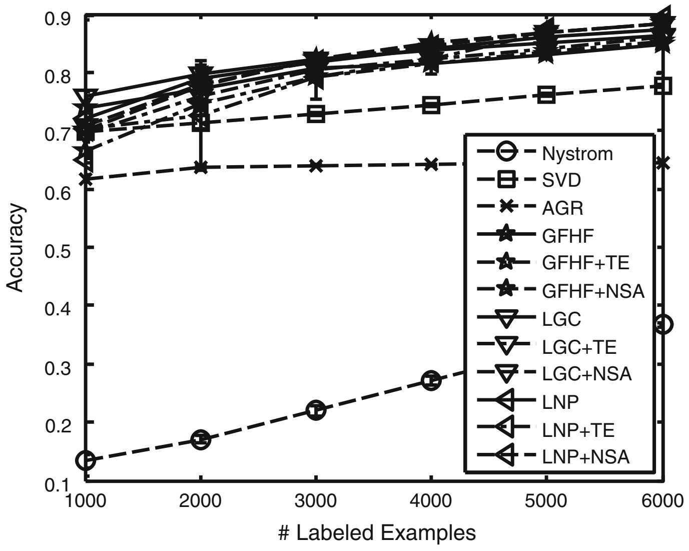

Fig. 4 Classification accuracies of various methods on *20 Newsgroups* dataset

Figure 3 reveals that the accuracies of all the algorithms can be improved by increasing the labeled examples, among which GFHF + NSA, LGC + NSA and LNP + NSA obtain better performance than AGR, SVD, TE and Nyström. Besides, we observe that the standard deviations of GFHF + NSA, LGC + NSA and LNP + NSA for different values of l are very small (see the short error bars), which demonstrates again that the approximation results are very robust to the choice of labeled examples.

Table 3 suggests that the implementations of original GFHF, LGC and LNP on *Isolet* dataset take very long time, while all the approximation methods are able to reduce the computational durations significantly. Comparatively, NSA and AGR are faster than SVD and Nyström, and their time costs remain substantially unchanged w.r.t the variation of labeled set. Therefore, NSA obtains the highest classification accuracy while only needs very little running time.

5.4 Text Classification

Text classification is an important application of machine learning techniques. *20 Newsgroups* dataset[5] is adopted to evaluate the performance of NSA and other baselines on text classification. This dataset is a collection of 18,846 newsgroups documents, which are partitioned into 20 different classes. TF-IDF feature [11], which is commonly adopted to represent text information, is utilized by us to characterize all documents. The parameters of baselines are $p = 50$ for SVD, $m = 80$ for AGR, and $c = 800$ and $r = 50$ for Nyström. With randomly selected labeled sets, we compare the accuracies and time expenditures of all algorithms mentioned above.

Figure 4 suggests that GFHF+NSA, LGC+NSA and LNP+NSA achieve around 80 % accuracies for different values of l, which is very close to the real results obtained by conventional GFHF, LGC and LNP. Comparatively, SVD, AGR and Nyström only achieve 70, 60 and 30 % accuracies approximately. Moreover, Table 4 reveals that inverting the large

[5] Avialable at http://qwone.com/~jason/20Newsgroups/.

Table 4 Time costs of different algorithms on *20 Newsgroups* dataset

Algorithms	Times (s)					
	$l = 1{,}000$	$l = 2{,}000$	$l = 3{,}000$	$l = 4{,}000$	$l = 5{,}000$	$l = 6{,}000$
GFHF	21826.3	21644.7	21701.3	21808.0	21983.8	21666.4
LGC	2663.7	2653.2	2655.1	2676.2	2646.0	2653.3
LNP	475.7	475.5	475.1	474.6	475.8	489.9
GFHF + TE	3729.2	3765.5	3818.8	3845.3	3809.2	3790.9
LGC + TE	3858.6	3813.3	4584.3	4701.9	4659.7	4632.0
LNP + TE	1148.7	1036.3	1069.7	1067.1	1073.3	1035.1
GFHF + NSA	6.9	7.0	7.0	7.0	7.2	7.1
LGC + NSA	7.2	7.0	7.1	7.1	7.0	7.0
LNP + NSA	7.7	7.8	7.7	7.8	7.8	7.7
Nyström	70.7	64.8	68.1	71.1	67.2	70.1
SVD	9.9	9.9	10.0	9.9	10.2	10.0
AGR	0.05	0.06	0.05	0.05	0.05	0.05

matrices in GFHF, LGC and LNP is extremely time-consuming. LGC + TE and LNP + TE even need much more computational time than the traditional LGC and LNP. Therefore, above algorithms are not feasible for large datasets like *20 Newsgroups*. However, when we apply NSA to deal with the inversion operation, the running time is reduced to about 7 s per conduction, which makes semi-supervised learning tractable for relatively large datasets. Note that AGR spends less time than NSA, but it cannot yield as high classification accuracy as NSA (see Fig. 4).

5.5 Complexity Analysis

From the above experiments, we clearly see that NSA saves considerable amount of running time for various GBSSL algorithms, including GFHF, LGC and LNP. NSA spends most of its time in computing the square of a sparse matrix, and its complexity is $O(s + s^2/n\ )$ with s denoting the number of non-zero elements in the sparse matrix. Comparatively, the original conduction of GFHF, LGC and LNP takes $O(n^3)$ complexity for inverting a huge $n \times n$ matrix. The total time complexities of AGR, SVD, TE and Nyström are $O(m^2 n)$, $O(ns)$, $O(\frac{1}{3}n^3)$ and $O(c^3 + ncr)$, respectively. Note that n is usually quite large while s is a very small number, so we can clearly see that the proposed NSA is able to achieve very competitive efficiency by comparing the complexities of various algorithms.

6 Conclusion

This paper proposes a simple yet effective method to enhance the scalability of a class of GBSSL algorithms. By observing that the spectral radius of matrix $\mathbf{P}$ is no more than 1, we adopt Neumann series to handle the inversion of a large sparse matrix. Our method is named as Neumann series approximation (NSA) and we have theoretically bounded the error between the results of NSA and direct inversion. Comprehensive experimental results suggest that NSA can successfully improve the scalability of some typical GBSSL algorithms by accelerating the running speed without losing much classification accuracy.

How to make other GBSSL algorithms such as manifold regularization [1] and measure propagation [15] practical for large scale data is still an open issue, and we will focus on this point for the future work.

Acknowledgments This research is supported by NSFC, China (No: 612732586137 5048), and Ph.D. Programs Foundation of Ministry of Education of China (No.20120073110018).

References

1. Belkin M et al (2006) Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J Mach Learn Res 7:2399–2434
2. Campbell Y, Davis T (1995) Computing the sparse inverse subset: an inverse multifrontal approach. Technical report TR-95-021, University of Florida, Gainesville, FL
3. Delalleau O, Bengio Y et al (2005) Efficient non-parametric function induction in semi-supervised learning. In: Proceedings of the 10th international workshop on artificial intelligence and statistics, p 12–19
4. Fergus R, Kandola J et al (2009) Semi-supervised learning in gigantic image collections. In: Proceedings of the advances in neural information processing systems, Vancouver
5. Fowlkes C et al (2004) Spectral grouping using the nyström method. Pattern Anal Mach Intell IEEE Trans 26(2):214–225
6. Garcke J, Griebel M (2005) Semi-supervised learning with sparse grids. In: Proceedings of the international conference on machine learning, Bonn
7. Golub G, Loan V (1996) Matrix computations. Johns Hopkins University Press, Baltimore
8. Kawano S et al (2012) Semi-supervised logistic discrimination via graph-based regularization. Neural Process Lett 36(3):203–216
9. Larsen R (1998) Lanczos bidiagonalization with partial reorthogonalization. DAIMI Report Series 27(537):1–101
10. Liu W, He J et al (2010) Large graph construction for scalable semi-supervised learning. In: Proceedings of the international conference on machine learning, Haifa, pp 679–686
11. Salton G, McGill M (1986) Introduction to modern information retrieval, McGraw-Hill, New York
12. Shang F et al (2012) Integrating spectral kernel learning and constraints in semi-supervised classification. Neural Process Lett 36(2):101–115
13. Sinha K, Belkin M (2009) Semi-supervised learning using sparse eigenfunction bases. Adv Neural Info Process Sys 22:1687–1695
14. Stewart G (1998) Matrix algorithms: basic decompositions. SIAM, Philadelphia
15. Subramanya A, Bilmes J (2011) Semi-supervised learning with measure propagation. J Mach Learn Res 12:3311–3370
16. Talwalkar A, Kumar S et al (2008) Large-scale manifold learning. In: Proceedings of the IEEE conference on the computer vision and pattern recognition (CVPR)
17. Tsang I, Kwok J (2007) Large-scale sparsified manifold regularization. In: Proceedings of the advances in neural information processing systems, Vancouver
18. Valls G et al (2007) Semi-supervised graph-based hyperspectral image classification. Geosci Remote Sens IEEE Trans 45(10):3044–3054
19. Wang J et al (2009) Linear neighborhood propagation and its applications. Pattern Anal Mach Intell IEEE Trans 31(9):1600–1615
20. Zhang K, Kwok J, et al (2009) Prototype vector machine for large scale semi-supervised learning. In: Proceedings of the international conference on machine learning, pp 1233–1240
21. Zhou D, Bousquet O (2003) Learning with local and global consistency. In: Proceedings of the advances in neural information processing systems, Vancouver, pp 321–328
22. Zhu X, Ghahramani Z et al (2003) Semi-supervised learning using Gaussian fields and harmonic functions. In: Proceedings of the international conference on machine learning, Washington, DC, pp 912–919
23. Zhu X, Goldberg B (2009) Introduction to semi-supervised learning. Morgan & Claypool Publishers, San Rafael
24. Zhu X (2005) Lafferty: Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In: Proceedings of the international conference on machine learning, Bonn, pp 1052–1059