Approximate Manifold Regularization: Scalable Algorithm and Generalization Analysis

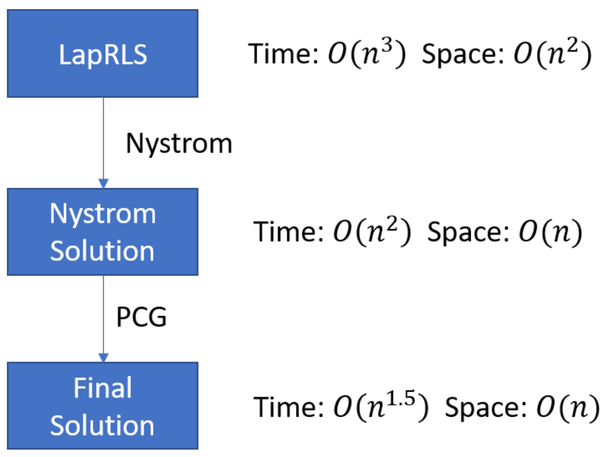
Jian Li, Yong Liu*, Rong Yin, and Weiping Wang
{lijian9026, liuyong, yinrong, wangweiping}@iie.ac.cn

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

Abstract

Graph-based semi-supervised learning is one of the most popular and successful semi-supervised learning approaches. Unfortunately, it suffers from high time and space complexity, at least quadratic with the number of training samples. In this paper, we propose an efficient graph-based semi-supervised algorithm with a sound theoretical guarantee. The proposed method combines Nystrom subsampling and preconditioned conjugate gradient descent, substantially improving computational efficiency and reducing memory requirements. Extensive empirical results reveal that our method achieves the state-of-the-art performance in a short time even with limited computing resources.

1. Core Idea : LapRLS + Nyström + PCG.



2. Contributions
(1) Scalable Algorithm with $\mathcal{O}(n)$ space and $\mathcal{O}(n^{1.5})$ time.
(2) Theoretical Guarantee: Excess risk bounds with convergence rate $\mathcal{O}(\frac{1}{\sqrt{m}})$.

Motivation

With squared loss function, the semi-supervised manifold regulariztion becomes LapRLS with a closed-form solution

$$\widehat{\boldsymbol{\alpha}} = (\mathbf{JK} + \lambda_A \mathbf{I} + \lambda_I \mathbf{LK})^{-1} \mathbf{y}_n, \qquad (1)$$

where $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ is $n \times n$ kernel matrix on train data, $\mathbf{J} = \text{diag}(1, \cdots, 1, 0, \cdots, 0)$ with the first m diagonal entries as 1 and the rest 0, and $\mathbf{y}_n = [y_1, y_2, \cdots, y_m, 0, \cdots, 0]^T$ with corresponding m labels and the rest filled by 0.
But LapRLS is not feasible to handle with large scale training data, due to its scalability issues:
(1) Space complexity: $\mathcal{O}(n^2)$.
(2) Time complexity: $\mathcal{O}(n^3)$.

LapRLS with Nyström

Consider a smaller hypothesis space $\mathcal{H}_s$

$$\mathcal{H}_s = \{f \in \mathcal{H} | f = \sum_{i=1}^{s} \alpha_i K(\mathbf{x}_i, \cdot), \boldsymbol{\alpha} \in \mathbb{R}^s\},$$

where $s \leq n$ and $\mathbf{x}_s = (\widetilde{\mathbf{x}}_1, \cdots, \widetilde{\mathbf{x}}_s)$ are Nyström centers selected by uniform subsampling from the training set. The solution of LapRLS (1) over the space $\mathcal{H}_s$ is in the form:

$$\boldsymbol{\alpha} = (\underbrace{\mathbf{K}_{ms}^T \mathbf{K}_{ms} + \lambda_A \mathbf{K}_{ss} + \lambda_I \mathbf{K}_{ns}^T \mathbf{L} \mathbf{K}_{ns}}_{\mathbf{H}})^{\dagger} \underbrace{\mathbf{K}_{ms}^T \mathbf{y}}_{\mathbf{z}},$$

where $\mathbf{H}^{\dagger}$ denotes the Moore-Penorse pseudoinverse, $(\mathbf{K}_{ms})_{ij} = K(\mathbf{x}_i, \widetilde{\mathbf{x}}_j)$ with $i \in \{1, \cdots, m\}$ and $j \in \{1, \cdots, s\}$, $(\mathbf{K}_{ss})_{kj} = K(\widetilde{\mathbf{x}}_k, \widetilde{\mathbf{x}}_j)$ with $k, j \in \{1, \cdots, s\}$ and $\mathbf{y} = [y_1, \cdots, y_m]^T \in \mathbb{R}^m$.

CG with Preconditioning

$$\mathbf{H}\boldsymbol{\alpha} = \mathbf{z} \quad \rightarrow \quad \mathbf{P}^{-1}\mathbf{H}\boldsymbol{\alpha} = \mathbf{P}^{-1}\mathbf{z}.$$

The number of iterations for preconditioning methods depends on the condition number $cond(\mathbf{P}^{-1}\mathbf{H})$, thus the preconditioner needs to be approximate to $\mathbf{H}$. To obtain a smaller condition number but also avoid inefficient computation, we define two preconditioners:
(1) $m \leq \sqrt{n}$

$$\mathbf{P} = \mathbf{K}_{ms}^T \mathbf{K}_{ms} + \lambda_A \mathbf{K}_{ss} + \frac{\lambda_I n^2}{s^2} \mathbf{K}_{ss} \mathbf{L}_{ss} \mathbf{K}_{ss}. \quad (2)$$

(2) $m > \sqrt{n}$

$$\mathbf{P} = \frac{m}{s} \mathbf{K}_{ss}^T \mathbf{K}_{ss} + \lambda_A \mathbf{K}_{ss} + \frac{\lambda_I n^2}{s^2} \mathbf{K}_{ss} \mathbf{L}_{ss} \mathbf{K}_{ss}. \quad (3)$$

To accelerate computation, $\mathbf{H}\boldsymbol{\alpha}$ is decomposed into a series of matrix-vector multiplications

$$\mathbf{H}\boldsymbol{\alpha} = \mathbf{K}_{ms}^T(\mathbf{K}_{ms}\alpha) + \lambda_A \mathbf{K}_{ss}\alpha + \lambda_I \mathbf{K}_{ns}^T(\mathbf{L}(\mathbf{K}_{ns}\alpha)). \quad (4)$$

Algorithm

Algorithm 1 Nyström LapRLS with PCG (`Nyström-PCG`)

Require: m labeled samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, $n - m$ unlabeled samples $\{\mathbf{x}_j\}_{j=m+1}^n$. Parameters: λ_A, λ_I, kernel method K and subsampling size s.
Ensure: coefficients $\boldsymbol{\alpha}$
1: Construct Laplacian graph matrix $\mathbf{L}$.
2: Select s Nyström centers with uniform sampling from training set $\{\widetilde{\mathbf{x}}_1, \cdots, \widetilde{\mathbf{x}}_s\} \in \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$.
3: Calculate inverse of the preconditioner $\mathbf{P}^{-1}$ (2) or (3).
4: Use any PCG solver to solve $\mathbf{P}^{-1}\mathbf{H}\boldsymbol{\alpha} = \mathbf{P}^{-1}\mathbf{z}$ with calculating $\mathbf{H}\boldsymbol{\alpha}$ by Eq. (4) and performing matrix multiplication in blocks.

Theoretical Analysis

Theorem 1 (Simple version)**.** *Under common assumptions and*

$$s \geq \mathcal{O}(\sqrt{n}) \quad \textit{and} \quad t \geq \mathcal{O}(\log m)$$

then the following excess risk bound holds with high probability,

$$\mathcal{E}(\widehat{f}_{\lambda,t}^s) - \mathcal{E}(f_{\mathcal{H}}) \leq \mathcal{O}(\frac{1}{\sqrt{m}}).$$

Technical challenges:

- Multi-penalty regularization. [1]
- Integral operator for Nyström. [2]
- Convergence of PCG. [3]

The complexity:

- Space complexity: $\mathcal{O}(s^2) = \mathcal{O}(n)$.
- Time complexity: $\mathcal{O}(nst + s^3 t) = \mathcal{O}(n\sqrt{n})$.

Estimators	Time	Space
RLS-Direct	$\mathcal{O}(m^3)$	$\mathcal{O}(m^2)$
LapRLS-Direct	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
LapRLS-CG	$\mathcal{O}(n^{2.5})$	$\mathcal{O}(n^2)$
LapRLS-PCG	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Nyström-Direct	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
Nyström-CG	$\mathcal{O}(n^{1.75})$	$\mathcal{O}(n)$
Nyström-PCG	$\mathcal{O}(n^{1.5})$	$\mathcal{O}(n)$

References

[1] Abhishake Rastogi and Sivananthan Sampath. Manifold regularization based on nystr {\" o} m type subsampling. *arXiv preprint arXiv:1710.04872*, 2017.

[2] Alessandro Rudi, Raffaello Camoriano, and Lorenzo Rosasco. Less is more: Nyström computational regularization. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 1657–1665, 2015.

[3] Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. Falkon: An optimal large scale kernel method. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 3888–3898, 2017.

Experimental Results

dataset	sample size	RLS-CG	LapRLS-CG	LapRLS-PCG	Nyström-CG	`Nyström-PCG`
space_ga	3107	1.251±0.004	**1.210±0.004**	**1.210±0.004**	**1.210±0.004**	**1.210±0.004**
phishing	11055	0.426±0.049	0.294±0.005	**0.273±0.007**	0.295±0.005	0.275±0.008
a8a	22696	0.702±0.002	**0.664±0.002**	**0.664±0.002**	0.664±0.002	0.664±0.002
w7a	24692	0.291±0.002	**0.283±0.002**	**0.283±0.002**	0.284±0.002	0.284±0.002
a9a	32561	0.698±0.005	**0.664±0.000**	**0.664±0.002**	**0.664±0.000**	**0.664±0.002**
ijcnn1	49990	0.434±0.005	**0.389±0.002**	**0.389±0.002**	0.393±0.001	0.463±0.001
cod-rna	59535	0.686±0.002	/	/	0.614±0.001	0.614±0.001
connect-4	67757	0.781±0.015	/	/	**0.739±0.002**	**0.739±0.002**
skin_nonskin	245057	3.119±0.023	/	/	**2.620±0.043**	**2.620±0.043**
YearPrediction	463715	0.198±0.001	/	/	**0.187±0.001**	**0.187±0.001**

Table 1: Comparison of average root mean square error between `Nyström-PCG` and RLS-CG, LapRLS-CG, LapRLS-PCG, Nyström LapRLS-CG. We bold the best results and underline the results of the other methods which are not significantly worse than the best one.

	RLS-CG		LapRLS-CG		LapRLS-PCG		Nyström-CG		`Nyström-PCG`	
	iter	time	iter	time	iter	time	iter	time	iter	time
space_ga	11	0.004	23	1.220	5	0.569	23	0.113	**2**	**0.016**
phishing	74	0.031	300	24.20	56	8.210	300	2.470	**3**	**0.045**
a8a	100	0.068	50	189.1	3	20.98	50	44.71	**1**	**4.370**
w7a	13	0.072	32	143.2	2	9.683	213	107.7	**1**	**2.252**
a9a	300	0.529	64	1699	3	30.30	65	70.40	**1**	**4.034**
ijcnn1	242	8.204	57	2154	9	72.41	53	108.8	**5**	**4.186**
cod-rna	96	7.178	/	/	/	/	55	134.6	**7**	**8.154**
connect-4	103	11.07	/	/	/	/	154	186.5	**10**	**4.220**
skin_nonskin	43	91.39	/	/	/	/	65	1490	**3**	**40.05**
YearPrediction	37	236.5	/	/	/	/	94	2479	**2**	**116.1**

Table 2: Comparison of average number of iterations and running time (seconds).