Fast and Scalable Graph-Based Semi-Supervised Learning

August 3, 2018

Abstract

We proposed a novel semi-supervised learning framework combining manifold learning, random projection, iterative solvers and preconditioning. Theoretical analysis shows our framework can achieve optimal statical accuracy with higher computation efficiency and less memory. Comparing with direct methods, the time complexity is reduced from $O(n^3)$ to $O(n\sqrt{n})$, and the memory complexity is reduced from $O(n^2)$ to $O(n)$. Therefore, large scale Semi-Supervised Learning can be solved quickly on limited resource under our framework.

1 Introduction

2 Notations and Preliminaries

2.1 Problem Definition

Assume there is a fixed but unknown distribution P on $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$. Specifically, l labeled examples $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\} \in \mathcal{X} \times \mathcal{Y}$ are drawn i.i.d from P and $n - l$ unlabeled examples $\{\mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\} \in \mathcal{X}$ are drawn i.i.d according to the marginal distribution $\mathcal{P}_X$ of P.

2.2 Graph-Based Semi-Supervised Learning

Manifold learning Methods based on Spectral Graph, known as Graph-Based SSL, is a typical solution to Semi-Supervised Learning [1, 4, 11]. Manifold learning is to find a smooth low-dimensional manifold embedded in the high-dimensional vector space, based on a set of sample points [2]. And Spectral graph theory studies eigenvector and eigenvalues of matrices which is generated by graph [5]. To ensure the solution is smooth with respect to both the ambient space and the marginal distribution $\mathcal{P}_X$, following manifold learning scheme is introduced

in [1]

$$f^* = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} \ell(y_i, f(\mathbf{x}_i)) + \lambda_A \|f\|_K^2 + \lambda_I \|f\|_I^2$$

where K is a Mercer kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and there is an associated RKHS $\mathcal{H}_K$. Intuitively, $\|f\|_I^2$ is an approximate penalty term that reflect the complexity of the function in *intrinsic* geometry of $\mathcal{P}_X$, while $\|f\|_K^2$ reflects the complexity of the function in the *abient* space.

When Marginal $\mathcal{P}_X$ is unknown, the optimization problem becomes

$$f^* = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} \ell(y_i, f(\mathbf{x}_i)) + \lambda_A \|f\|_K^2 + \lambda_I \int_{\mathbf{x} \in \mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 d\mathcal{P}_X(\mathbf{x})$$

where $\mathcal{M}$ is compact submanifold $\mathcal{M} \in \mathbb{R}^n$.

Based on different definitions of similarity between two points, there are two kinds of Graph-Based SSL: global methods and local methods. Global methods calculate connections between all pair points. But local methods only consider connections between neighbourhood points. In this paper, we apply the widely used local method called Laplacian Eigenmaps, which is to find a map to keep local geometry features of points on average. By choosing exponential weights of the adjacency graph, there is convergence of the graph Laplacian to the Laplace-Beltrami operator on the manifold

$$f^* = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} \ell(y_i, f(\mathbf{x}_i)) + \lambda_A \|f\|_K^2 + \frac{\lambda_I}{n^2} \mathbf{f}^T \mathbf{L} \mathbf{f} \tag{1}$$

where $\mathbf{L}$ is the graph Laplacian matrix by $\mathbf{L} = \mathbf{D} - \mathbf{W}$ and $\mathbf{f} = [f(\mathbf{x}_1), \cdots, f(\mathbf{x}_n)]^T$. Here, $\mathbf{W} \in \mathbb{R}^{n \times n}$ records undirected weight between points and the diagonal matrix $\mathbf{D}$ is given by $D_{ii} = \sum_{j=1}^{n} W_{ij}$.

Theorem 1. *The Represent Theorem was given in [1], which can be used to expansion of kernel functions in terms of the labeled and unlabeled data*

$$f^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \tag{2}$$

where K is a positive definite kernel.

2.3 Laplacian Regularized Least Squares

When the loss function ℓ use the squared loss, the optimization problem in (1) becomes

$$\arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} (y_i - f(\mathbf{x}_i))^2 + \lambda_A \|f\|_K^2 + \frac{\lambda_I}{n^2} \mathbf{f}^T \mathbf{L} \mathbf{f} \tag{3}$$

Applying the Represent Theorem, we arrive at a convex differentiable objective function of $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_n]^T$

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{l}(\mathbf{Y} - \mathbf{JK}\boldsymbol{\alpha})^T(-\mathbf{JK})) + \lambda_A \boldsymbol{\alpha}^T \mathbf{K}\boldsymbol{\alpha} + \frac{\lambda_I l}{n^2}\boldsymbol{\alpha}^T \mathbf{KLK}\boldsymbol{\alpha}$$

Setting the derivation of the objective function be zero leads following solution, which reduce computations to a linear system

$$(\mathbf{JK} + \lambda_A l\mathbf{I} + \frac{\lambda_I l}{n^2}\mathbf{LK})\boldsymbol{\alpha}^* = \mathbf{Y} \tag{4}$$

Computations. For large datasets, it is a challenge to solve Equation (4) in a direct approach, which needs $O(n^2)$ in space and $O(n^2 d + n^3)$ in time.

2.4 Random Projection

To accelerate large scale kernelized Semi-Supervised Learning, we can use random subsampling to get a approximate matrix $\mathbf{K}_{ns}$ instead of primal kernel matrix $\mathbf{K}_{nn}$. There are two kinds of popular random projection methods: Nyström [12, 10] and random features [6]. In this work, we focus on a basic Nyström approach based on following form

$$\tilde{f}^s_\lambda(\mathbf{x}) = \sum_{i=1}^{s} \tilde{\alpha}_i K(\mathbf{x}_i, \mathbf{x}), \qquad \text{with} \quad \{\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_s\} \in \{\mathbf{x}_1, \cdots, \mathbf{x}_l, \mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\}$$

defined only s train examples sampled uniformly from whole n examples. Then, only s coefficients are needed in following linear system derived from (3) based on Nyström approach

$$\tilde{f}^s_\lambda = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l}\sum_{i=1}^{l}(y_i - \mathbf{K}_{ls(i)}\boldsymbol{\alpha})^2 + \lambda_A \boldsymbol{\alpha}^T \mathbf{K}_{ss}\boldsymbol{\alpha} + \frac{\lambda_I}{n^2}(\mathbf{K}_{ns}\boldsymbol{\alpha})^T \mathbf{L}(\mathbf{K}_{ns}\boldsymbol{\alpha})$$

Let the derivation of the objective function be zero, we can get

$$\frac{2}{l}\mathbf{K}^T_{ls}(\mathbf{K}_{ls}\boldsymbol{\alpha} - \mathbf{y}) + 2\lambda_A \mathbf{K}_{ss}\boldsymbol{\alpha} + \frac{2\lambda_I}{n^2}\mathbf{K}^T_{ns}\mathbf{LK}_{ns}\boldsymbol{\alpha} = \mathbf{0}$$

Then, we can get a linear system

$$(\mathbf{K}^T_{ls}\mathbf{K}_{ls} + \lambda_A l\mathbf{K}_{ss} + \frac{\lambda_I l}{n^2}\mathbf{K}^T_{ns}\mathbf{LK}_{ns})\boldsymbol{\alpha} = \mathbf{K}^T_{ls}\mathbf{y} \tag{5}$$

Computations. Direct method solving equation (2.5) needs $O(ns)$ to store $\mathbf{K}_{ns}$ in space while it needs only $O(s^2)$ memory if $\mathbf{K}^T_{ls}\mathbf{K}_{ls}$ and $\mathbf{K}^T_{ns}\mathbf{LK}_{ns}$ are computed in blocks of dimension at most $s \times s$, and $O(ns^2)$ for computation.

Statistics. As long as $s \gg n$, random projection can dramatically reduce memory cost. However, a question arises of whether this comes at expenses of statistical accuracy and of how many random examples is needed. Recent work shows that $s = O(\sqrt{n})$ is enough to achieve statistical accuracy.

2.5 Precondition and Gradient Methods

The condition number is the radio between the largest and smallest singular value of matrix defined in the problem [9], while it can capture the time complexity of iteratively solving the corresponding linear system. For a basic iterative method to solve (2.5)

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \tau[(\mathbf{K}_{ls}^T\mathbf{K}_{ls} + \lambda_A l\mathbf{K}_{ss} + \frac{\lambda_I l}{n^2}\mathbf{K}_{ns}^T\mathbf{L}\mathbf{K}_{ns})\boldsymbol{\alpha} - \mathbf{K}_{ls}^T\mathbf{y}] \tag{6}$$

By the above gradient decent method, the number of iterations needed for ϵ accurate solution of problem (2.5)

$$t = O(cond(\mathbf{K}_{ls}^T\mathbf{K}_{ls} + \lambda_A l\mathbf{K}_{ss} + \frac{\lambda_I l}{n^2}\mathbf{K}_{ns}^T\mathbf{L}\mathbf{K}_{ns})\log(1/\epsilon)).$$

It is shown in [3] that $t = O(\sqrt{n})$ can achieve good statistical properties. However, preconditioning can accelerate iterative methods by using a suitable matrix B to get better condition number. For (2.5), we can get approximate matrix $\mathbf{B}$ by

$$\mathbf{B}\mathbf{B}^T \approx (\mathbf{K}_{ls}^T\mathbf{K}_{ls} + \lambda_A l\mathbf{K}_{ss} + \frac{\lambda_I l}{n^2}\mathbf{K}_{ns}^T\mathbf{L}\mathbf{K}_{ns})^{-1} = H^{-1}$$

and $\mathbf{B}^T\mathbf{H}\mathbf{B}\boldsymbol{\beta} = \mathbf{B}^T\mathbf{y}$. Clearly, $\boldsymbol{\alpha}_* = \mathbf{B}\boldsymbol{\beta}_*$ is solution of (2.5).

We can use following form to approximate H and B is the Cholesky decomposition of H^{-1}, and the approximate is based on random subsampling which can be Nyström approximation.

$$\boldsymbol{B}\boldsymbol{B}^T = (\frac{l}{l'}\mathbf{K}_{l's}^T\mathbf{K}_{l's} + \lambda_A l\mathbf{K}_{ss} + \frac{\lambda_I l(n+s)}{2ns^2}\mathbf{K}_{ss}^T\mathbf{L}_{ss}\mathbf{K}_{ss})^{-1} \tag{7}$$

where l' is the number of subsampling examples from labeled data and $l' < s$. The above preconditioning is a natural approximation of the ideal preconditioning of problem () that is $\boldsymbol{B}\boldsymbol{B}^T = H^{-1}$. Specifically, of last part, $\mathbf{K}_{ns}$ becomes $\mathbf{K}_{ss}$ by subsamping uniformly from n to s which is partly from labeled data and partly from unlabeled data.

Computation. Obviously, we can use (7) and $\mathbf{B}^T\mathbf{H}\mathbf{B}\boldsymbol{\beta} = \mathbf{B}^T\mathbf{y}$ to solve linear system (2.5) via conjugate gradient [9] known as CG. Thus, our method needs $O(nst + s^3)$ in time and $O(s^2)$ in space.

3 Algorithms

Our approach is combination of approximation, iterative method and preconditioning, which consists two level approximation: (1) Approximation on primal problem, resulting linear system (2.5), (2) Approximation on preconditioning, resulting preconditioner (7). Finally, solving the problem by preconditioner and CG.

Algorithm 1 Fast and Scalable Semi-Supervised Learning

Require: l labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $n-l$ unlabeled examples $\{\mathbf{x}_j\}_{j=l+1}^{n}$. Parameters: λ_A, λ_I, kernel method K and subsampling size s.

Ensure: coefficients $\boldsymbol{\alpha}$

1: Construct adjacent graph by k-NN or graph kernel with all n nodes.
2: Define edge weight $W_{ij} = \exp^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/4t}$ by heat kernel.
3: Compute graph Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$ where $\mathbf{D}$ is a diagonal matrix given by $D_{ii} = \sum_{j=1}^{n} W_{ij}$.
4: Compute kernel matrix $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$
5: Subsample uniformly on primal problem to get $\mathbf{K}_{ns}$ by Nyström
6: Calculate matrix $\mathbf{H} = \mathbf{K}_{ls}^T \mathbf{K}_{ls} + \lambda_A l \mathbf{K}_{ss} + \frac{\lambda_I l}{n^2} \mathbf{K}_{ns}^T \mathbf{L} \mathbf{K}_{ns}$
7: Compute preconditioner $\tilde{\mathbf{H}} = (\frac{l}{l'} \mathbf{K}_{l's}^T \mathbf{K}_{l's} + \lambda_A l \mathbf{K}_{ss} + \frac{\lambda_I l(n+s)}{2ns^2} \mathbf{K}_{ss}^T \mathbf{L}_{ss} \mathbf{K}_{ss})$ via subsampling uniformly by Nyström
8: Compute $\mathbf{B}\mathbf{B}^T = \tilde{\mathbf{H}}^{-1}$ with Cholesky decomposition of preconditioner by Nyström
9: Solve $\mathbf{B}^T \mathbf{H} \mathbf{B} \boldsymbol{\beta} = \mathbf{B}^T \mathbf{y}$ via Conjugate Gradient.
10: Compute $\boldsymbol{\alpha}_* = \mathbf{B} \boldsymbol{\beta}_*$

4 Convergence and Complexity Analysis

Firstly, statistical analysis about Nyström method on Graph-based Semi-Supervised Learning are given in [7]. According to its **Theorem 3.1**, we can see the convergence rate is $O(n^{-\frac{br}{2br+b+1}})$, which can easily get convergence rate at $O(\frac{1}{\sqrt{n}})$. Meanwhile, convergence about approximation on preconditioner is analysed in [8], whose proof of **Theorem 3** shows approximation on preconditioner also can achieve optimal statistical properties. Analysis in [7, 8] shows at least $s = O(\sqrt{n})$ subsampling examples and $t = O(\log n)$ iterations can give good theoretical guarantee. Thus, our method needs only $O(n)$ space and $O(n\sqrt{n})$ time to get optimal statistical accuracy.

More details will be added in future.

5 Empirical Study

References

[1] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[2] William M Boothby. *An introduction to differentiable manifolds and Riemannian geometry*, volume 120. Academic press, 1986.

[3] Raffaello Camoriano, Tomás Angles, Alessandro Rudi, and Lorenzo Rosasco. Nytro: When subsampling meets early stopping. In *Artificial Intelligence and Statistics*, pages 1403–1411, 2016.

[4] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[5] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[6] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

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

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

[9] Yousef Saad. *Iterative methods for sparse linear systems*, volume 82. siam, 2003.

[10] Alex J Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. 2000.

[11] Amarnag Subramanya and Partha Pratim Talukdar. Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(4):1–125, 2014.

[12] Christopher KI Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.