Improved Large-Scale Graph Learning through Ridge Spectral Sparsification

Daniele Calandriello[1,2], Ioannis Koutis[3], Alessandro Lazaric[4], Michal Valko[1]

[1]SequeL team, INRIA Lille - Nord Europe. [2] LCSL-IIT/MIT. [3] New Jersey Institute of Technology. [4] Facebook AI Research Paris.

Motivation

Graphs are ubiquitous (e.g. Facebook $n = 10^9, m = 10^{12}$)
↳ typical graph algorithm has $\mathcal{O}(mn)$ time and $\mathcal{O}(m)$ space cost

Large graphs do not fit in a single machine memory

Hard to solve with engineering:
↳ multiple passes slow, distribution has communication costs

Hard to solve for natural graphs (i.e. no vectorial representation)
↳ sparsity level cannot be chosen

Make the graph sparse, while preserving its spectral structure

Already known in graph community: spectral graph sparsifiers
↳ but ML models also have regularization

1) Can we reduce memory costs without reducing accuracy?
2) Does regularization help us to further reduce memory costs?
3) Can we do so without assumptions and increased runtime?

Learning on Graphs

The graph $\mathcal{G} = (\mathcal{X}, \mathcal{E})$ is undirected and weighted

- $|\mathcal{X}| = n$ nodes and $|\mathcal{E}| = m$ edges
- The weights $a_{e_{i,j}}$ encodes the distance between nodes

The Laplacian of $\mathcal{G}$ is the PSD matrix $\mathbf{L}_\mathcal{G} \triangleq \mathbf{D}_\mathcal{G} - \mathbf{A}_\mathcal{G}$.

- Using edge-indicator vector $\mathbf{b}_e \triangleq \sqrt{a_e}(\chi_i - \chi_j)$
  ↳ $\mathbf{L}_\mathcal{G} = \sum_e \mathbf{b}_e\mathbf{b}_e^\mathsf{T} = \mathbf{B}_\mathcal{G}^\mathsf{T}\mathbf{B}_\mathcal{G}$ Positive Semi-Definite
- $\mathcal{G}$ is connected, $\mathbf{L}_\mathcal{G}$ has only one 0 eigenvalue and $\mathrm{Ker}(\mathbf{L}_\mathcal{G}) = \mathbf{1}$

Laplacian smoothing (LapSmo) with Gaussian noise.
Let $\mathbf{y} \triangleq \mathbf{f}^\star + \xi$ be a noisy measurement of $\mathbf{f}^\star$ with $[\xi]_i \sim \mathcal{N}(0, \sigma^2)$.

$$\widehat{\mathbf{f}} \triangleq \arg\min_{\mathbf{f}\in\mathbb{R}^n} (\mathbf{f}-\mathbf{y})^\mathsf{T}(\mathbf{f}-\mathbf{y}) + \lambda\mathbf{f}^\mathsf{T}\mathbf{L}_\mathcal{G}\mathbf{f} = (\lambda\mathbf{L}_\mathcal{G} + \mathbf{I})^{-1}\mathbf{y}, \qquad (1)$$

where λ is a regularization parameter.

Graph semi-supervised learning (SSL).

- There exists a label y_i for each node in $\mathcal{G}$
- $\mathcal{S}$ is the set of l labeled nodes
- $\mathcal{T}$ is the set of $u = n - l$ unlabeled nodes
- $\mathbf{I}_\mathcal{S} \in \mathbb{R}^{n\times n}$ is the diagonal indicator matrix of nodes in $\mathcal{S}$
- $\mathbf{C} = c_l\mathbf{I}_\mathcal{S} + c_u\mathbf{I}_\mathcal{T}$ and $c_l \geq c_u > 0$
- $\mathbf{y}_\mathcal{S} \triangleq \mathbf{I}_\mathcal{S}\mathbf{y} \in \mathbb{R}^n$
- With input $\mathcal{X}$, $\mathcal{S}$ and $\mathbf{y}_\mathcal{S}$, return a labeling $\mathbf{f} \in \mathbb{R}^n$

harmonic function solution (HFS):

$$\begin{aligned}\widehat{\mathbf{f}}_{\texttt{HFS}} &\triangleq \arg\min_{\mathbf{f}\in\mathbb{R}^n} \tfrac{1}{l}(\mathbf{f}-\mathbf{y})^\mathsf{T}\mathbf{I}_\mathcal{S}(\mathbf{f}-\mathbf{y}) + \lambda\mathbf{f}^\mathsf{T}\mathbf{L}_\mathcal{G}\mathbf{f} \\ &= (\lambda l\mathbf{L}_\mathcal{G} + \mathbf{I}_\mathcal{S})^+\mathbf{y}_\mathcal{S}. \end{aligned} \qquad (2)$$

stable harmonic function solution (STA):

$$\begin{aligned}\widehat{\mathbf{f}}_{\texttt{STA}} &= \arg\min_{\mathbf{f}\in\mathbb{R}^n} \tfrac{1}{l}(\mathbf{f}-\mathbf{y})^\mathsf{T}I_\mathcal{S}(\mathbf{f}-\mathbf{y}) + \lambda\mathbf{f}^\mathsf{T}L_\mathcal{G}\mathbf{f} + \tfrac{\mu}{l}\mathbf{f}^\mathsf{T}\mathbf{1} \\ &= (\lambda lL_\mathcal{G} + I_\mathcal{S})^+\left(\mathbf{y}_\mathcal{S} - \frac{\mathbf{y}_\mathcal{S}^\mathsf{T}(\lambda lL_\mathcal{G} + I_\mathcal{S})^+\mathbf{1}}{\mathbf{1}^\mathsf{T}(\lambda lL_\mathcal{G} + I_\mathcal{S})^+\mathbf{1}}\mathbf{1}\right). \end{aligned} \qquad (3)$$

local transductive regression solution (LTR):

$$\begin{aligned}\widehat{\mathbf{f}}_{\texttt{LTR}} &\triangleq \arg\min_{\mathbf{f}\in\mathbb{R}^n} (\mathbf{f}-\mathbf{y})^\mathsf{T}\mathbf{C}(\mathbf{f}-\mathbf{y}) + \mathbf{f}^\mathsf{T}(\mathbf{L}_\mathcal{G} + \lambda\mathbf{I})\mathbf{f} \\ &= (\mathbf{C}^{-1}(\mathbf{L}_\mathcal{G} + \lambda\mathbf{I}) + \mathbf{I})^{-1}\mathbf{y}_\mathcal{S}. \end{aligned} \qquad (4)$$

Spectral clustering (SC).

$$\widehat{\mathbf{F}} \triangleq \arg\min_{\mathbf{F}:\mathbf{F}^\mathsf{T}\mathbf{F}=\mathbf{I}_k, \mathbf{f}_c\perp\mathbf{1}} \mathrm{Tr}(\mathbf{F}^\mathsf{T}\mathbf{L}\mathbf{F}).$$

Near-Linear Time Solvers

Pseudo-inverse $\mathbf{L}_\mathcal{G}^+$ dense
↳ $\mathcal{O}(n^3)$ time to construct and $\mathcal{O}(n^2)$ space to store

Use iterative method (e.g. GD) to solve $\|\mathbf{L}_\mathcal{G}\mathbf{x} - \mathbf{y}\|^2$:
↳ $\mathcal{O}(m)$ space, $\mathcal{O}(mt)$ time, $\lambda_{\max}(\mathbf{L}_\mathcal{G})/\lambda_{\min}(\mathbf{L}_\mathcal{G}) \simeq n$ iter.

Preconditioned Conjugate GD + recursive sparsification
↳ $\mathcal{O}(m)$ space, $\mathcal{O}(m\log(n))$ time, $\lambda_{\max}(\mathbf{L}_\mathcal{G})/\lambda_{\min}(\mathbf{L}_\mathcal{G}) \simeq 1$ iter.

Cost of learning on graph: $\mathcal{O}(m)$ space/time, $\mathcal{O}(\log(n))$ passes

References

[1] Belkin, M., Matveeva, I., and Niyogi, P.. Regularization and Semi-Supervised Learning on Large Graphs. In *COLT*, 2004.

[2] Cortes, C., Mohri, M., Pechyony, D., and Rastogi, A.. Stability of transductive regression algorithms. In *ICML*, 2008.

[3] Koutis, Ioannis, Miller, Gary L., and Peng, Richard. A nearly-m log n time solver for SDD linear systems. In *FOCS*, 2011.

[4] Calandriello, D., Lazaric, A., and Valko, M. Distributed sequential sampling for kernel matrix approximation. In *ICML*, 2017.

[5] Cohen, M., Musco, C., and Musco, C. Input sparsity time low-rank approximation via ridge leverage score sampling. In *SODA*, 2017.

[6] Fergus, Rob, Weiss, Yair, and Torralba, Antonio. Semi-Supervised Learning in Gigantic Image Collections. In *NIPS*, 2009.

[7] Kelner, J. A. and Levin, A.. Spectral Sparsification in the Semi-streaming Setting. *Theory of Comput. Syst.*, 53(2):243–262, 2013.

Ridge Spectral Sparsifiers

Definition 1. A (ε,γ)-spectral sparsifier of $\mathcal{G}$ is a re-weighted sub-graph $\mathcal{H} \subseteq \mathcal{G}$ whose Laplacian $\mathbf{L}_\mathcal{H}$ satisfies

$$(1-\varepsilon)\mathbf{L}_\mathcal{G} - \varepsilon\gamma\mathbf{I} \preceq \mathbf{L}_\mathcal{H} \preceq (1+\varepsilon)\mathbf{L}_\mathcal{G} + \varepsilon\gamma\mathbf{I}. \qquad (5)$$

Spectrum is preserved with mixed multiplicative/additive error

$$(1-\varepsilon)\lambda_i(\mathbf{L}_\mathcal{G}) - \varepsilon\gamma \leq \lambda_i(\mathbf{L}_\mathcal{H}) \leq (1+\varepsilon)\lambda_i(\mathbf{L}_\mathcal{G}) + \varepsilon\gamma,$$

Preserves all directions larger than γ
An $(\varepsilon, 0)$-spectral sparsifier is a traditional ε-sparsifier

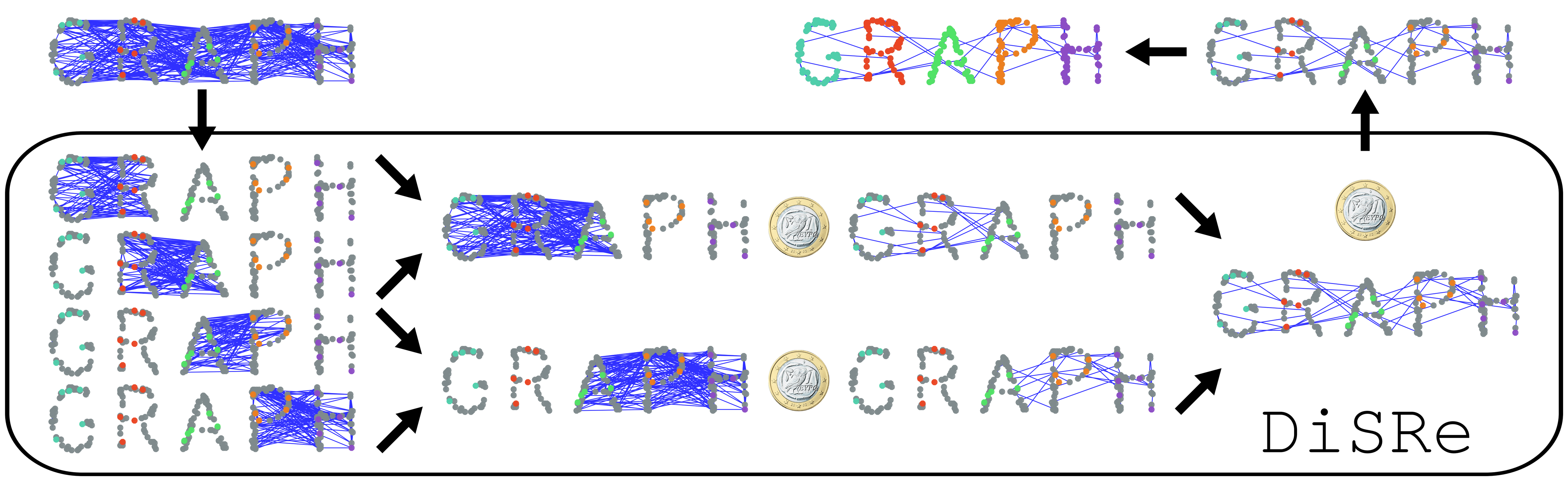
Definition 2. Given a graph $\mathcal{G}$, define
γ-effective resistance: $r_e(\gamma) = \mathbf{b}_e^\mathsf{T}(\mathbf{B}_\mathcal{G}^\mathsf{T}\mathbf{B}_\mathcal{G} + \gamma\mathbf{I})^{-1}\mathbf{b}_e$

Effective dimension: $d_{\text{eff}}(\gamma) = \sum_e r_e(\gamma) = \sum_{i=1}^n \frac{\lambda_i(\mathbf{L}_\mathcal{G})}{\lambda_i(\mathbf{L}_\mathcal{G}) + \gamma} \leq n$

Proposition 1 ([5] (informal)). Starting from the empty graph, construct $\mathcal{H}$ by adding each edge in $\mathcal{G}$ to $\mathcal{H}$ independently with probability $p_e = \overline{q}r_e(\gamma)$. If $\overline{q} \geq 4\log(4n/\delta)/\varepsilon^2$, then w.p. $1-\delta$, $\mathcal{H}$ is an (ε,γ)-sparsifier with $\mathcal{O}(d_{\text{eff}}(\gamma)\overline{q})$ edges.

Computing $r_e(\gamma)$ requires $\mathcal{O}(m)$ time/space and multiple passes over the graph. Can we do better?

Distributed Sequential Resparsification



Algorithm 1 The DiSRe algorithm.

1: **Input:** $\mathcal{G}$ ε, γ, δ, **Output:** (ε,γ)-sparsifier $\mathcal{H}_\mathcal{G}$
2: Partition $\mathcal{G}$ into k sub-graphs:
  $\mathcal{H}_{1,l} \leftarrow \mathcal{G}_l \leftarrow \{(e_{i,j}, \widetilde{p}_{1,e} = 1)\}$
3: Initialize set $\mathcal{S}_1 = \{\mathcal{H}_{1,l}\}_{l=1}^k$
4: **for** $h = 1, \dots, k-1$ **do**
5:   Pick two sparsifiers $\mathcal{H}_{h,i'}, \mathcal{H}_{h,i'}$ from $\mathcal{S}_h$
6:   $\overline{\mathcal{H}} \leftarrow$ `Merge-Resparsify`$(\mathcal{H}_{h,i}, \mathcal{H}_{h,i'})$
7:   Place $\overline{\mathcal{H}}$ back into $\mathcal{S}_{h+1}$
8: **end for**
9: Return $\mathcal{H}_\mathcal{G}$, the last sparsifier in $\mathcal{S}_k$

Algorithm 2 `Merge-Resparsify`

1: **Input:** (ε,γ)-sparsifiers $\mathcal{H}_{h,i}, \mathcal{H}_{h,i'}$ of graphs $\mathcal{G}_{h,i}, \mathcal{G}_{h,i'}$
2: **Output:** $\overline{\mathcal{H}}$, an (ε,γ) sparsifier of $\mathcal{G}_{h,i} + \mathcal{G}_{h,i'}$
3: Initialize $\overline{\mathcal{H}} = \mathcal{H}_{h,i} + \mathcal{H}_{h,i'}$
4: For all $e \in \overline{\mathcal{H}}$, use a fast SDD solver to compute

$$\widetilde{r}_{h+1,e}(\gamma) \leftarrow (1-\varepsilon)\mathbf{b}_e^\mathsf{T}(\mathbf{L}_{\overline{\mathcal{H}}} + (1+\varepsilon)\gamma\mathbf{I})^{-1}\mathbf{b}_e$$

5: Set probabilities $\widetilde{p}_{h+1,e} \leftarrow \min\{\overline{q}\widetilde{r}_{h+1,e}(\gamma), \widetilde{p}_{h,e}\}$
6: Sample $z_{h+1,e}$ from Bernoulli$(\widetilde{p}_{h+1,e}/\widetilde{p}_{h,e})$ (i.e. coin-flip)
7: Return $\overline{\mathcal{H}} \leftarrow \{(e_{i,j}, \widetilde{p}_{h+1,e})\}$ for all $z_{h+1,e} \neq 0$

Theorem 1. Let $\varepsilon > 0$ be the accuracy, $0 \leq \delta \leq 1$ the probability of error, and $\rho \triangleq (1+3\epsilon)/(1-\epsilon)$. Given an arbitrary graph $\mathcal{G}$ and an arbitrary merge tree structure, if DiSRe is run with over-sampling parameter $\overline{q} \triangleq 26\rho\log(3n/\delta)/\varepsilon^2$, then with probability $1-\delta$

(1) each sub-graphs $\mathcal{H}_{\{h,l\}}$ is an (ε,γ)-sparsifier of $\mathcal{G}_{\{h,l\}}$
(2) with at most $3\overline{q}d_{\text{eff}}(\gamma)$ edges.

Space: $\mathcal{O}(d_{\text{eff}}(\gamma)\log(n)) \leq \mathcal{O}(n\log(n))$ (independent from m)

Time: $\mathcal{O}(d_{\text{eff}}(\gamma)\log^3(n))$ for fully balanced and $k \geq m/(d_{\text{eff}}(\gamma)\overline{q})$
↳ With only $\mathcal{O}(m\log^3(n))$ work (loading $\mathcal{G}$ is $\Omega(m)$)

Communication: only $\mathcal{O}(\log(n))$ rounds
↳ removed edges are forgotten single pass/streaming
  ↳ point-to-point, centralization only to choose tree

SSL with DiSRe

Setting. The labels are bounded $|\mathbf{y}(x)| \leq c$ and $\mathcal{F}$ is the set of centered functions such that $|\mathbf{f}(x) - \mathbf{y}(x)| \leq 2c$.

Theorem 2. If the labels $\widetilde{\mathbf{y}}_S$ are centered then, w.p. $1-\delta$, $\widehat{\mathbf{f}}_{\texttt{STA}}$ computed on a $(\varepsilon,0)$-sparsifier $\mathcal{H}$ satisfies

$$R(\widetilde{\mathbf{f}}) \leq \widehat{R}(\widehat{\mathbf{f}}) + \beta + \left(2\beta + \frac{c^2(l+u)}{lu}\right)\sqrt{\frac{\pi(l,u)\ln\frac{1}{\delta}}{2}} + \frac{1+\varepsilon}{1-\varepsilon}\left(\frac{2\varepsilon l\gamma\lambda_2(\mathbf{L}_\mathcal{G})c}{((1-\varepsilon)l\gamma\lambda_2(\mathbf{L}_\mathcal{G})-1)^2}\right)^2,$$

$$\beta \leq \frac{3c\sqrt{l}}{((1-\varepsilon)l\gamma\lambda_2(\mathbf{L}_\mathcal{G})-1)^2} + \frac{4c}{(1-\varepsilon)l\gamma\lambda_2(\mathbf{L}_\mathcal{G})-1}, \quad \pi(l,u) \triangleq \frac{lu}{l+u-0.5}\,\frac{2\max\{l,u\}}{2\max\{l,u\}-1}$$

$\mathcal{O}(n\log(n))$ space, $\mathcal{O}(n\log^3(n))$ time
↳ $\mathcal{O}(m\log^3(n))$ work
  ↳ EXACT $\mathcal{O}(m)$ time/space

Preserve risk rate: only $\frac{1+\varepsilon}{1-\varepsilon}$ slower

First bound without assumptions on $\mathcal{G}$
↳ require centered $\widetilde{\mathbf{f}}$

LapSmo with DiSRe

Theorem 3. Let $\widehat{\mathbf{f}}$ be the LapSmo solution computed using $\mathbf{L}_\mathcal{G}$ and $\widetilde{\mathbf{f}}$ the solution computed using its (ε,γ)-sparsifier $\mathbf{L}_\mathcal{H}$. Then,

$$\|\widehat{\mathbf{f}} - \widetilde{\mathbf{f}}\|_2^2 \leq \frac{\varepsilon^2}{1-\varepsilon}(0.25 + \lambda\gamma)\left(\lambda\widehat{\mathbf{f}}^\mathsf{T}\mathbf{L}_\mathcal{G}\widehat{\mathbf{f}} + \lambda\gamma\|\widehat{\mathbf{f}}\|_2^2\right).$$

$\mathcal{O}(d_{\text{eff}}(\gamma)\log(n))$ space, $\mathcal{O}(d_{\text{eff}}(\gamma)\log^3(n))$ time
↳ exploit regularization: $\mathcal{H}$ sub-linear in n

In general, requires $\gamma \propto \widehat{\mathbf{f}}^\mathsf{T}\mathbf{L}_\mathcal{G}\widehat{\mathbf{f}}/\|\widehat{\mathbf{f}}\|$
↳ trade-off between smoothness and decay of $\mathbf{L}_\mathcal{G}$

Experiments

Dataset: Amazon co-purchase graph from `https://snap.stanford.edu/data/com-Amazon.html` (Yang and Leskovec, 2012)
↳$n = 334,863$ nodes, natural, artificially sparse (true graph known only to Amazon)
  ↳we compute 4-step random walk to recover removed co-purchases, $m = 98,465,352$ edges (294 avg. degree)

Target: For LapSmo $\mathbf{v}$ eigenvector associated with smallest eigenvalue of $\mathbf{L}_\mathcal{G}$, for SSL sign$(\mathbf{v})$.

Alg.	*Parameters*	$\lvert\mathcal{E}\rvert$ *(x10*^{6}*)*	*Err.* SSL (l=346)	*Err.* SSL (l=672)	*Err.* $D(\widetilde{\mathbf{f}})(\sigma=10^{-3})$	*Err.* $D(\widetilde{\mathbf{f}})$ $(\sigma=10^{-2})$
EXACT		98.5	0.312 ± 0.022	0.286 ± 0.010	0.067 ± 0.0004	0.756 ± 0.006
kN	$k=60$	15.7	0.329 ± 0.0143	0.311 ± 0.027	0.172 ± 0.0004	0.822 ± 0.002
kN	$k=90$	21.2	0.334 ± 0.024	0.311 ± 0.024	0.125 ± 0.0002	0.811 ± 0.003
DiSRe	$\gamma=0$, $\overline{q}=100$	15	0.314 ± 0.0165	**0.296** ± 0.015	0.068 ± 0.0003	0.758 ±0.005
DiSRe	$\gamma=0$, $\overline{q}=150$	22.8	**0.314** ± 0.0158	0.310 ± 0.024	0.068 ± 0.0004	**0.756** ± 0.005
DiSRe	$\gamma=10^3$, $\overline{q}=100$	7.3	–	–	0.072 ± 0.0003	0.789 ± 0.005
DiSRe	$\gamma=10^2$, $\overline{q}=100$	11.8	–	–	**0.068** ± 0.0002	0.772 ± 0.004
DiSRe	$\gamma=10$, $\overline{q}=100$	14.4	–	–	0.068 ± 0.0004	0.760 ± 0.004

Time: Loading $\mathcal{G}$ from disk 90s, DiSRe 720s($k = 4 \times 8$ CPU) - 120s($k = 4 \times 32$ CPU), kN 60s, computing $\widetilde{\mathbf{f}}$ 120s, computing $\widehat{\mathbf{f}}$ 720s
Space EXACT and $\mathcal{G}$ 30GB, DiSRe or kN and $\mathcal{H}$ 10GB