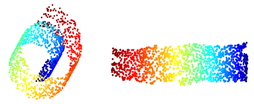
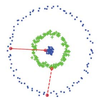
Fast Graph Laplacian Regularized Kernel Learning via Semidefinite-Quadratic-Linear Programming

Xiao-Ming Wu, Anthony Man-Cho So, Zhenguo Li, and Shuo-Yen Robert Li
The Chinese University of Hong Kong

Background



Dimensionality Reduction



Constrained Clustering

Kernel Learning Framework

Semidefinite Programming

SDP is not scalable to large-scale problems!

The Problem

How to speed up SDP?

Low-rank Kernel Approximation

$$K = QYQ^T \quad Q \in \mathbb{R}^{n\times m} \quad (m \ll n)$$

Q consists of the smoothest m eigenvectors of graph Laplacian.

Convex Quadratic Semidefinite Program (QSDP)

$$\min_{\mathbf{y}} \ \mathbf{y}^T A\mathbf{y} + \mathbf{b}^T\mathbf{y}$$
$$\text{s.t. } Y \succeq 0$$

How to solve this QSDP?

Previous Approaches

By Schur Complement

$$(A^{\frac{1}{2}}\mathbf{y})^T(A^{\frac{1}{2}}\mathbf{y}) \le \nu$$

Schur Complement Based SDP (SCSDP)

$$\min_{\mathbf{y},\nu} \ \nu + \mathbf{b}^T\mathbf{y}$$

This LMI is very large!

$$\text{s.t. } Y \succeq 0 \text{ and } \begin{pmatrix} I_{m^2} & A^{\frac{1}{2}}\mathbf{y} \\ (A^{\frac{1}{2}}\mathbf{y})^T & \nu \end{pmatrix} \succeq 0$$

Any better formulation?

Our Formulation

Semidefinite-Quadratic-Linear Programming (SQLP)

Step 1: Cholesky Factorization

$$\min_{\mathbf{y}} \ \mathbf{y}^T A\mathbf{y} + \mathbf{b}^T\mathbf{y}$$
$$\text{s.t. } Y \succeq 0$$

$$A = B^T B \quad B \in \mathbb{R}^{r\times m^2}$$

$$\min_{\mathbf{y},\mathbf{z},\mu} \ \mu + \mathbf{b}^T\mathbf{y}$$
$$\text{s.t. } \mathbf{z} = B\mathbf{y},$$
$$\mathbf{z}^T\mathbf{z} \le \mu,$$
$$Y \succeq 0.$$

Step 2: Let $\mathbf{u} = (\frac{1+\mu}{2}, \frac{1-\mu}{2}, \mathbf{z}^T)^T$

$C = (\mathbf{0}_{r\times 2}, I_{r\times r})$

$$\min_{\mathbf{y},\mathbf{z},\mu} \ \mu + \mathbf{b}^T\mathbf{y}$$
$$\text{s.t. } \mathbf{z} = B\mathbf{y},$$
$$\mathbf{z}^T\mathbf{z} \le \mu,$$
$$Y \succeq 0.$$

$$\min_{\mathbf{y},\mathbf{u}} \ (\mathbf{e}_1 - \mathbf{e}_2)^T\mathbf{u} + \mathbf{b}^T\mathbf{y}$$
$$\text{s.t. } (\mathbf{e}_1 + \mathbf{e}_2)^T\mathbf{u} = 1,$$
$$B\mathbf{y} - C\mathbf{u} = \mathbf{0},$$
$$\mathbf{u} \in \mathcal{K}_{r+2},$$
$$Y \succeq 0,$$

Second-order cone:

$$\mathcal{K}_n = \{(x_0; \mathbf{x}) \in \mathbb{R}^n : \ x_0 \ge \|\mathbf{x}\|\}$$

Speedup Gain

SCSDP: A large LMI **vs.** **SQLP:** A second-order cone; Some *linear* equalities

Speedup gain is of the order $m^{2.5}$

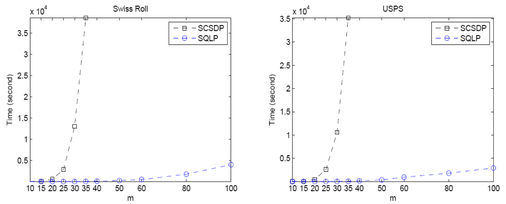


Figure 2: Curves on computational cost: m vs. Time.

Experimental Results

Table 1: Computational Results on Swiss Roll (Time: second; # Iter: number of iterations)

	SCSDP			SQLP			
m	Time	# Iter	Time per Iter	Time	# Iter	Time per Iter	rank(A)
10	3.84	29	0.13	0.96	32	0.03	64
15	60.36	30	2.01	1.75	31	0.06	153
20	557.79	32	17.43	4.48	35	0.13	264
25	2821.76	34	82.99	7.84	37	0.21	403
30	13039.30	37	352.41	13.39	35	0.38	537
35	38559.50	33	1168.50	29.74	35	0.85	670
40	>1 day	—	—	74.01	35	2.12	852
50	—	—	—	213.26	35	6.09	1152
60	—	—	—	467.90	35	13.37	1451
80	—	—	—	1729.65	39	44.35	2062
100	—	—	—	3988.31	36	110.79	2623

Table 2: Computational Results on USPS (Time: second; # Iter: number of iterations)

	SCSDP			SQLP			
m	Time	# Iter	Time per Iter	Time	# Iter	Time per Iter	rank(A)
10	2.84	21	0.14	0.47	16	0.03	100
15	42.96	22	1.95	1.26	17	0.07	225
20	461.38	27	17.09	3.35	17	0.20	400
25	2572.72	31	82.99	5.97	14	0.43	625
30	10576.01	30	352.53	15.72	19	0.83	900
35	35173.60	30	1172.50	44.53	17	2.62	1225
40	>1 day	—	—	133.58	20	6.68	1600
50	—	—	—	362.24	16	22.64	2379
60	—	—	—	936.58	19	49.29	2938
80	—	—	—	1784.12	17	104.95	3112
100	—	—	—	2900.44	17	170.61	3111