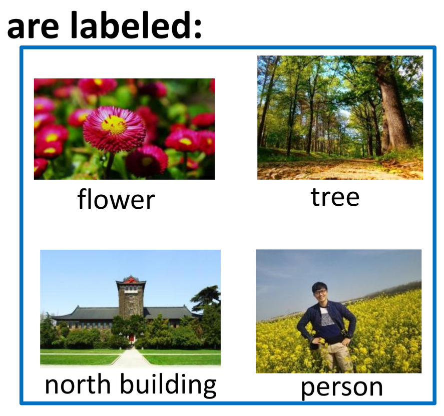
Storage Fit Learning with Unlabeled Data

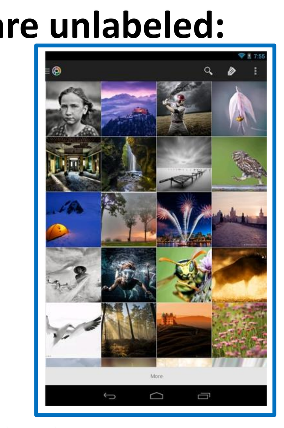
Bo-Jian Hou, Lijun Zhang, Zhi-Hua Zhou

Motivation 1

- Smart mobile devices are widely used and can generate large amount of data.
- Face the classification task on mobile devices.

Few photos are labeled: Most photos are unlabeled:



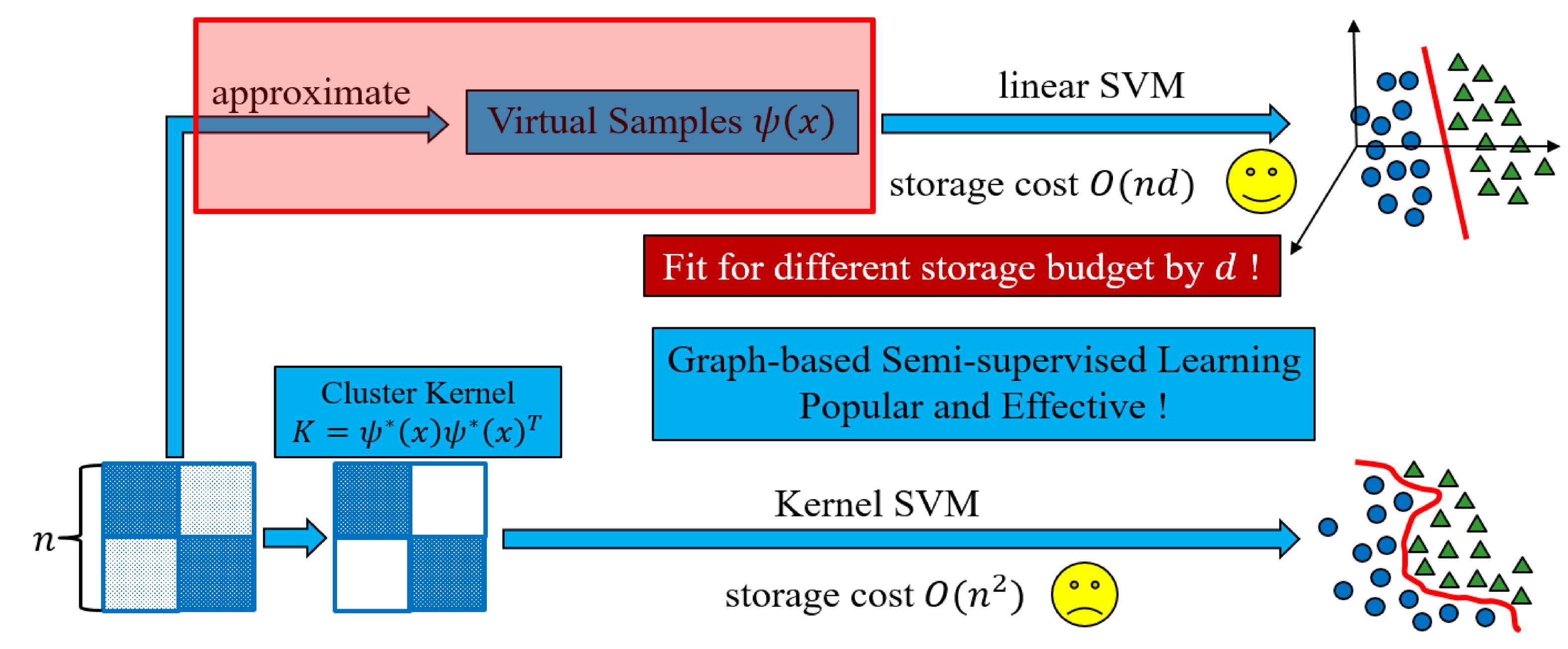


With labeled and unlabeled data, we can do semi-supervised learning.

- The storage for the learning process on mobile devices is **limited**.
- Different mobile devices have **different** limited memories:

We need to adjust the semi-supervised algorithms to fit for different storage budgets.

Basic Idea 2



- **Obtaining "virtual samples" to transform kernel SVM to linear SVM;**
- **Then storage costs from $O(n^2)$ to $O(nd)$;**
- **Fit for different storage budgets by d !**

Proposed Methods 3

Now we are aimed at obtaining "virtual samples".

cluster kernel

$$\tilde{K} = \tilde{D}^{1/2}\tilde{L}\tilde{D}^{1/2}$$

virtual samples

$$F = \tilde{K}^{1/2} = \tilde{D}^{1/2}\tilde{L}^{1/2}$$

$$\tilde{L}^{1/2} \approx D^{-1/2}U_k(\Sigma_k)^{1/2} \qquad \tilde{D}_{ii} = 1/\tilde{L}_{ii}$$

top k eigenvector of original kernel

modified top k eigenvalue of original kernel

We only need to find the eigensystem of the original kernel !

Two methods to find the eigensystem (eigenvalues and eigenvectors) of L or K:

Stochastic Optimization for Cluster Kernel (SoCK)

To obtain the top eigensystem of L, we need to find a low-rank matrix $\hat{L}$ to approximate L.

Singular Value Thresholding (SVT)

$$\hat{L} = \mathcal{D}_\lambda[L] = \sum_{i:\sigma_i \geq \lambda} (\sigma_i - \lambda)\boldsymbol{u}_i\boldsymbol{u}_i^\top$$

Stochastic Composite Optimization (SCO)

$$\min_{Z \in \mathbb{R}^{n\times n}} \frac{1}{2}\|Z - L\|_F^2 + \lambda\|Z\|_*$$

L_ξ is generated by Random Fourier Features

$$\min_{Z \in \mathbb{R}^{n\times n}} \frac{1}{2}E[\|Z - L_\xi\|_F^2] + \lambda\|Z\|_*$$

SPGD to solve it and take the last iteration as the final solution

$$Z_{t+1} = \mathcal{D}_{\eta_t\lambda}\left[(1-\eta_t)Z_t + \eta_t L_t\right]$$

Then we need to do SVD decomposition on $(1-\eta_t)Z_t + \eta_t L_t$ where L_t and Z_t can be split into two matrices: $L_t = \zeta_t \chi_t^\top \ \zeta_t, \chi_t \in \mathbb{R}^{n\times a_t}$, $Z_t = U_t V_t^\top U_t, V_t \in \mathbb{R}^{n\times b_t}$.

So in each iteration we only need to do SVD decomposition on $[\sqrt{(1-\eta_t)}U_t, \sqrt{\eta_t}\zeta_t]$.

Space complexity $O(n(a_t + b_t))$ where a_t and b_t is much smaller than n, through adjusting b_t, we can fit for different storage budgets.

Nystrom Cluster Kernel (NysCK)

To obtain the top eigensystem of K, we sample instances and use Nystrom to directly calculate the eigensystem of K.

$$K = \begin{bmatrix} W & K_{21}^\top \\ K_{21} & K_{22} \end{bmatrix} \quad \text{and} \quad C = \overbrace{\begin{bmatrix} W \\ K_{21} \end{bmatrix}}^{s} \qquad W_k = U_{W,k}\Sigma_{W,k}U_{W,k}^\top$$

SVD decomposition

$$\underbrace{\Sigma_k}_{\text{eigenvalues}} = \left(\frac{n}{s}\right)\Sigma_{W,k} \quad \text{and} \quad \underbrace{U_k}_{\text{eigenvectors}} = \sqrt{\frac{s}{n}}CU_{W,k}\Sigma_{W,k}^\dagger$$

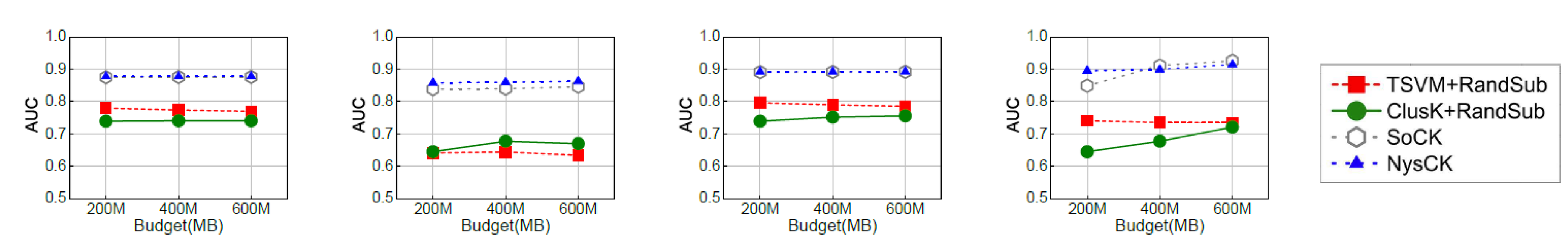
Space complexity $O(ns)$ where s is much smaller than n, through adjusting s, we can fit for different storage budgets.

Space and time complexity comparisons

Methods	Space	Time
ClusK	$O(n^2)$	$O(n^3)$
NysCK	$O(n(d+k))$	$O(n(nd+k^2))$
SoCK	$O(n(d+a_t+b_t))$	$O(n[nd+da_t+(a_t+b_t)^2+k^2])$

Experiment 4

With Storage Budget:



(a) 1K labeled training examples @adult-a (b) 1K labeled training examples @w8a (c) 3K labeled training examples @adult-a (d) 3K labeled training examples @w8a

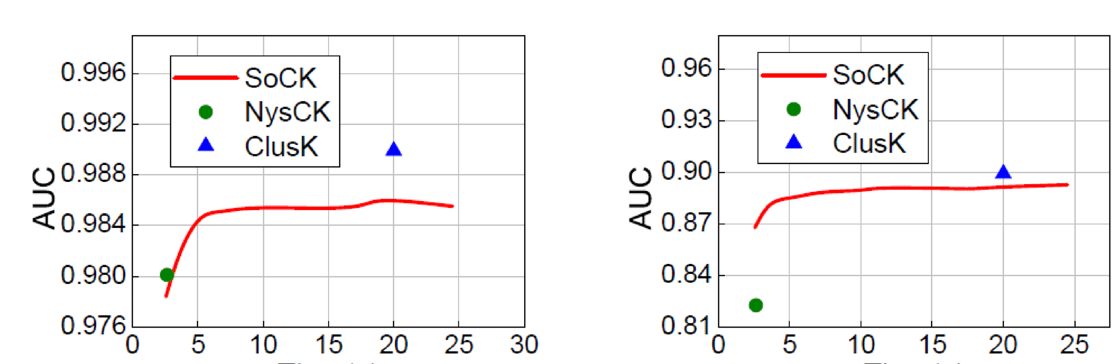
SoCK and NysCK outperform other two methods under all budgets.

Without Storage Budget:

Dataset [size, dim]	KNN	Harmonic	CMN	TSVM	ClusK	SoCK	NysCK
australian[690,42]	.743(7)	.754(5)	.754(6)	.851(4)	.873(3)	**.902**(1)	.897(2)
credit-a[653,15]	.805(7)	.874(5)	.864(6)	.894(3)	**.901**(1)	.884(4)	.896(2)
credit-g[1000,20]	.591(7)	.670(5)	.670(6)	.700(4)	.711(2)	**.723**(1)	.705(3)
diabetes[768,8]	.640(7)	.737(5)	.737(6)	.781(2)	**.801**(1)	.775(3)	.757(4)
german[1000,59]	.587(7)	.665(4)	.665(5)	.655(6)	.691(2)	**.710**(1)	.669(3)
kr-vs-kp[3196,36]	.821(7)	.917(6)	.917(5)	.928(4)	**.990**(1)	.985(2)	.980(3)
splice[3175,60]	.678(7)	.782(5)	.782(6)	.825(3)	**.899**(1)	.891(2)	.823(4)
svmguide3[1284,22]	.605(7)	.645(4)	.643(5)	.629(6)	.769(2)	.701(3)	**.771**(1)
Total rank	56	39	45	32	13	17	22

The proposed algorithms achieve competitive performance on all data sets.

SoCK vs NysCK:



- **NysCK gets an approximate solution with a not high AUC value in a short time.**
- **SoCK can refine its solution continuously with the decreasing of approximation error and outperforms NysCK after a few seconds.**
- **SoCK is more effective while NysCK is more efficient.**

Conclusion 5

- A new setting: storage fit learning with unlabeled data.
- Key: given different storage budgets, the behavior of the algorithm should be adjusted differently.
- Concern algorithms relying on spectral analysis which suffer seriously from storage burden of kernel matrix.
- Utilize the techniques of low-rank approximation to adapt these algorithms to fit for a given storage budget.

Learning And Mining from DatA
http://lamda.nju.edu.cn

Contact

houbj@lamda.nju.edu.cn; zhanglj@lamda.nju.edu.cn; zhouzh@lamda.nju.edu.cn.