A study of kernel SVM approximation methods

DC-Pred++ and LDKL

Dhruv Singal[1] Pranav Maneriker[1]

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

CS678A Learning with Kernels, Fall 2015

Outline

1. Introduction
2. DC-Pred++
   - Prior Art
   - Algorithm
3. LDKL
   - Prior Art
   - Description
4. Performance comparison

Introduction

- Kernel methods increases the range of application for standard algorithms like SVM, Ridge Regression, PCA etc.
- Difficult to calculate and store the kernel matrix for all samples:
  - Kernel calculation time: $\mathcal{O}(n^2 d)$
  - Space: $\mathcal{O}(n^2)$
  - Prediction time: $\mathcal{O}(\bar{n}d)$, with $\bar{n} = \#\text{SVs}$
- Approximation of kernel matrix increases time and space efficiency of the kernel SVM algorithm
- LDKL [1] and DC-Pred++ [2] are state of the art in approximate kernel SVM methods

Outline

1. Introduction
2. DC-Pred++
   - Prior Art
   - Algorithm
3. LDKL
   - Prior Art
   - Description
4. Performance comparison

Nyström Method

History

- Nyström method for approximating Gram Matrix introduced by Williams and Seeger, 2001 [3]
- Great improvements made by a series of papers like Drineas and Mahoney, 2005 [4]
- Kumar et al., 2009 proposed an ensemble model of Nyström approximations to achieve state of the art [5]

Nyström Method

A brief peek

Given $m << n$ *landmark* points, $\{\boldsymbol{u}_j\}_{j=1}^{m}$, the Nyström method forms $C \in \mathbb{R}^{n \times m}$ and $W \in \mathbb{R}^{m \times m}$ such that $C_{ij} = K(\boldsymbol{x}_i, \boldsymbol{u}_j)$ and $W_{ij} = K(\boldsymbol{u}_i, \boldsymbol{u}_j)$ to get

$$G \approx \bar{G} = CW^{\dagger}C^{T} \tag{1}$$

with a nice bound on $\|\bar{G} - G\|_{\xi},\ \xi = 2, F$ [4]
The decision value is calculated as

$$\boldsymbol{c}(W^{\dagger}C^{T}\boldsymbol{\alpha}) = \boldsymbol{c}\boldsymbol{\beta} \tag{2}$$

where, $\boldsymbol{c} = [K(\boldsymbol{x}, \boldsymbol{u}_1), \ldots, K(\boldsymbol{x}, \boldsymbol{u}_m)]$

Nyström Method

Pros and Cons

- Kernel calculation time: $\mathcal{O}(m^2 n)$
- Space: $\mathcal{O}(mn)$
- Prediction time: $\mathcal{O}(md)$

Nyström Method

Pros and Cons

- Kernel calculation time: $\mathcal{O}(m^2 n)$
- Space: $\mathcal{O}(mn)$
- Prediction time: $\mathcal{O}(md)$

Typically, $m > 100$ needed for reasonable accuracy [2].

	Prediction Time	Approximation Error
Large m	$\uparrow$	$\downarrow$
Small m	$\downarrow$	$\uparrow$

How to resolve this trade off?

Outline

1. Introduction
2. DC-Pred++
   - Prior Art
   - Algorithm
3. LDKL
   - Prior Art
   - Description
4. Performance comparison

Main Idea

Three novel propositions:

- Add *pseudo-landmark points* to resolve the tradeoff
- Use *weighted k-means* to achieve better bounds
- Use *divide and conquer* approach for better prediction time

Pseudo landmark points

- Add p pseudo landmark points $\{\boldsymbol{v}_t\}_{t=1}^{p}$ from $\mathbb{R}^d$ (not necessary input samples)
- Estimate $K(\boldsymbol{x}, \boldsymbol{v}_t)$ for all t using a function $f_t : \mathbb{R}^m \longrightarrow \mathbb{R}$ using $\boldsymbol{c} = [K(\boldsymbol{x}, \boldsymbol{u}_1), \ldots, K(\boldsymbol{x}, \boldsymbol{u}_m)]$ as the input for all f_t
- f_t is obtained by:
  - Triangle inequality for stationary kernels
  - Regression and (low degree) polynomial basis functions for general kernels

Pseudo landmark points

- Obtain $\bar{C} = [C, C']$ by augmenting the matrix C with the estimated values of $K(\boldsymbol{x}_i, \boldsymbol{v}_t)$ using f_t, giving

$$G \approx \bar{G} = \bar{C}\bar{W}\bar{C}^T \text{ and } \bar{W} = \bar{C}^{\dagger} G (\bar{C})^T \tag{3}$$

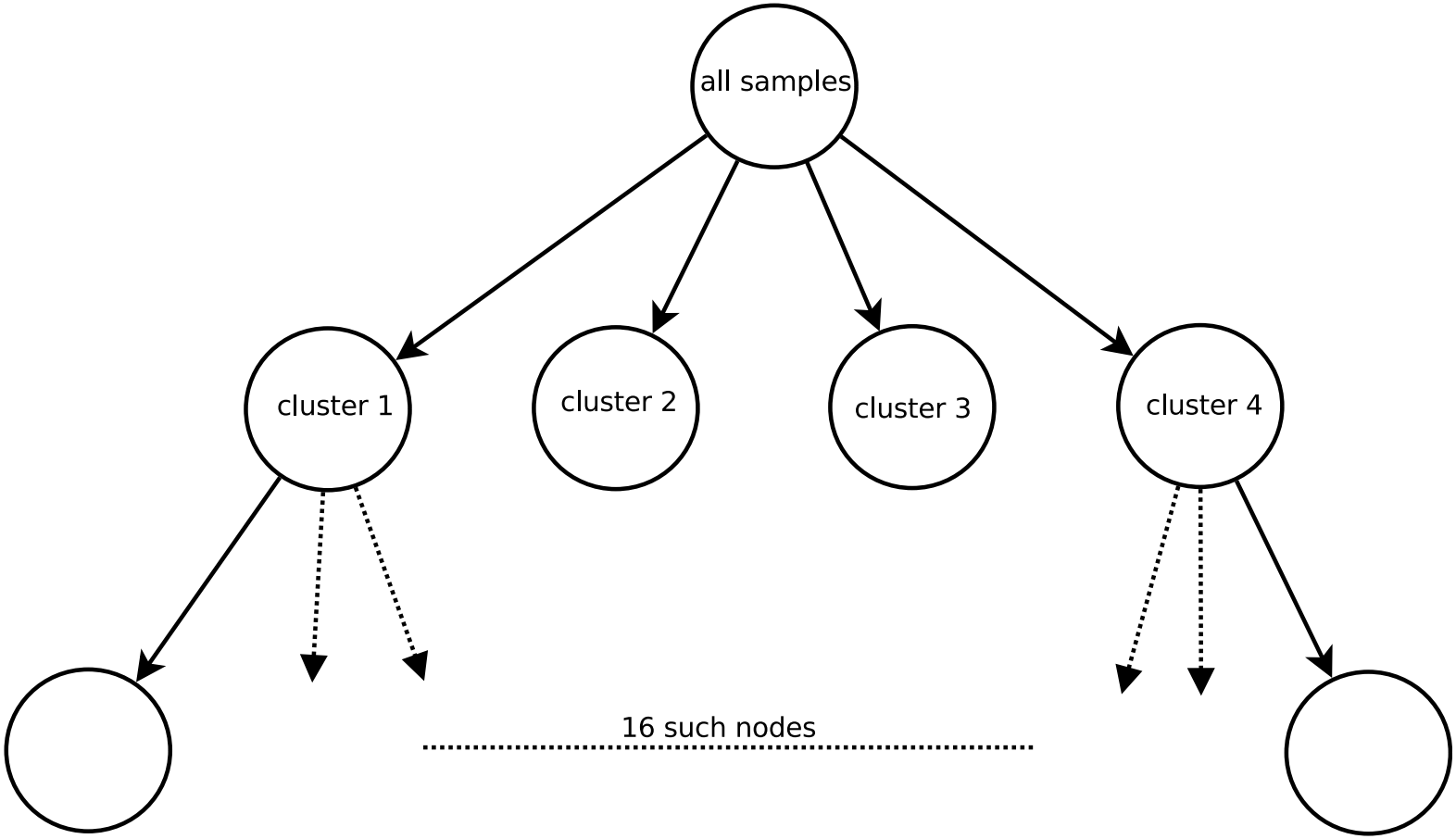
- Instead of G in the RHS, use a submatrix of the kernel matrix while minimizing approximation error

Weighted k-means

- It suffices to minimize kernel approximation error on $\{i\}$ with large $|\alpha_i^*|$, instead of all samples
- For stationary kernels, the following gives minimum error:
  - Perform weighted k-means using $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as distance measure and $\{\alpha_i^*\}_{i=1}^n$ as weights
  - For all p clusters, the cluster centroids are the pseudo landmark points
  - Use any approximate solver to get the weights $\{\alpha_i^*\}_{i=1}^n$

Divide and conquer

- Modified the approach taken by Hsieh et al, 2013 [6]
- Use k-means on input space distances to form a hierarchial clustering



Divide and conquer

- Assign the pseudo landmark points obtained by weighted k-means to nearest clusters
- Train a Nyström approximation model on every cluster, using these local pseudo landmark points
- *Early prediction*: Return the prediction of the local cluster model, instead of the global model, for the test sample as in [6]

Note: Divide and conquer and Weighted k-means applicable for SVM and Ridge Regression only

Outline

1. Introduction
2. DC-Pred++
   - Prior Art
   - Algorithm
3. LDKL
   - Prior Art
   - Description
4. Performance comparison

Multiple Kernel Learning

History

- Studies such as Lanckriet et al., 2004 show that combining multiple kernels improves classification performance [7]
- Simplest implementation: Unweighted sum - Pavlidis et al., 2001, Ben-Hur and Noble 2005 [8, 9]
- Bach et al., 2004 showed a method of incorporating SMO in convex combination of kernels [10]

Multiple Kernel Learning

Localized Multiple Kernel Learning (LMKL)

Assigning different weights to kernels in different regions may improve classification accuracy

LMKL [11] is an important landmark towards the development of LDKL utilizing this idea

LMKL

$$y(\mathbf{x}) = sign(\sum_k p(\boldsymbol{w}_k|\mathbf{x})\mathbf{w}_k^t\phi_k(\mathbf{x}) + b) \tag{4}$$

$$p(\mathbf{w}_k|\mathbf{x}) = \frac{e^{\boldsymbol{\theta}_k^t\boldsymbol{x}+\theta_{0k}}}{\sum_m e^{\boldsymbol{\theta}_m^t\boldsymbol{x}+\theta_{0m}}} \tag{5}$$

$$\Theta = \{(\boldsymbol{\theta}_k, \theta_{0k})\} \tag{6}$$

Solver based on the efficient MKL solver in Rakotomamonjy et al., [12]

Localized Multiple Kernel Learning

Pros and Cons

Experimental results show:

- Distinct kernels
  - Accuracy unchanged, support vectors $\downarrow$
- Same kernels
  - Accuracy $\uparrow$, support vectors $\downarrow$

Outline

1. Introduction
2. DC-Pred++
   - Prior Art
   - Algorithm
3. LDKL
   - Prior Art
   - Description
4. Performance comparison

Localized Deep Kernel Learning

The LDKL[1] learns a non-linear kernel K as a product of a global and a local kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = K_L(\boldsymbol{x}_i, \boldsymbol{x}_j) K_G(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

LDKL

$$y(\mathbf{x}) = sign(W^t(\boldsymbol{x})\phi_G(\boldsymbol{x})) \tag{7}$$

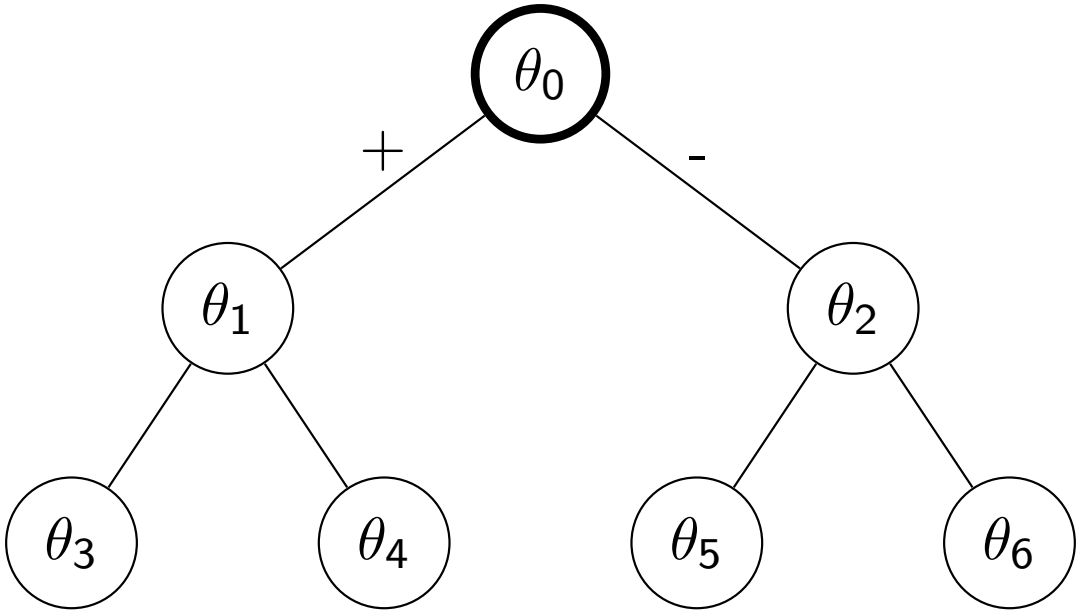
$$\boldsymbol{w}_k = \sum_i \alpha_i y_i \phi_{L_k}(\boldsymbol{x}_i)\phi_G(\boldsymbol{x}_i), \phi_L \in \mathbb{R}^M \tag{8}$$

$$W = [\boldsymbol{w}_1, ..., \boldsymbol{w}_M] \tag{9}$$

$$W(\boldsymbol{x}) = W\phi_L(\boldsymbol{x}) \tag{10}$$

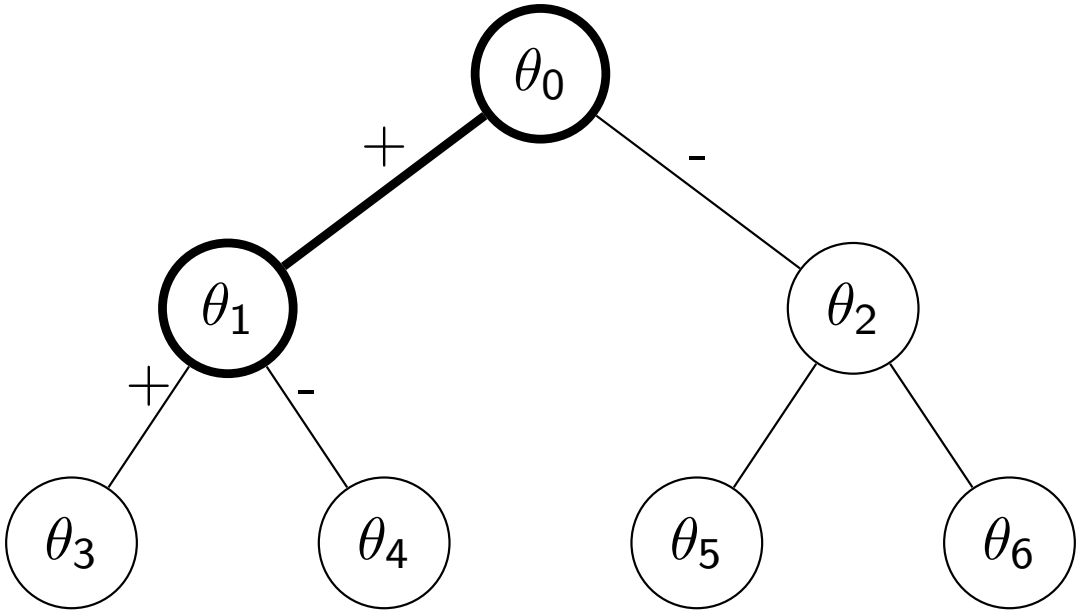
Local Kernel Tree

$$\boldsymbol{\theta}_0^t \boldsymbol{x} > 0$$



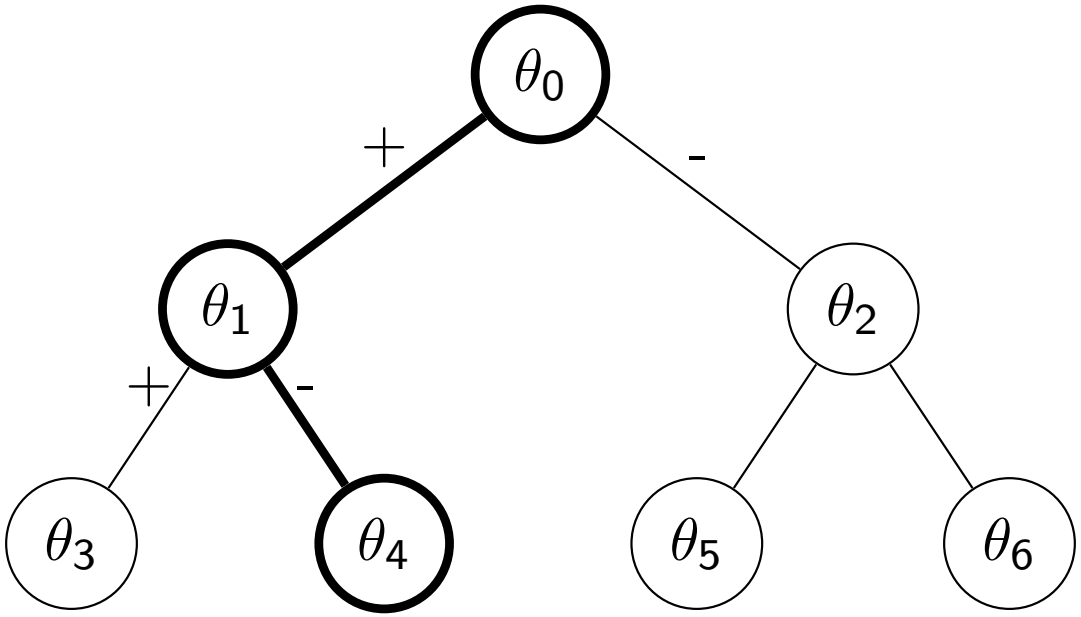
Local Kernel Tree

$$\boldsymbol{\theta}_0^t \boldsymbol{x} > 0, \boldsymbol{\theta}_1^t \boldsymbol{x} < 0$$



Local Kernel Tree

$$\boldsymbol{\theta}_0^t \boldsymbol{x} > 0, \boldsymbol{\theta}_1^t \boldsymbol{x} < 0, \boldsymbol{\theta_4}$$



Local Kernel

For a deep representation of tree like structured local kernel, we choose

$$\phi_{L_k}(\boldsymbol{x}) = I_k(\boldsymbol{x}) f_{k_0}(\boldsymbol{x}, f_{k_1}(\boldsymbol{x}, ...(f_{k_R}(\boldsymbol{x}, 1)))) \tag{11}$$

where each k_i is the i^{th} ancestor of k and

$$I_k(\boldsymbol{x}) = \Pi_{I \in Ancestors(k)} \frac{1}{2}(sign(\boldsymbol{\theta}_I^t \boldsymbol{x}) + (-1)^{C(I)}) \tag{12}$$

$C(I) = 0$ if node I is its parents left child and $C(I) = 1$ if it is its parents right child

Local Kernel

For a deep representation of tree like structured local kernel, we choose

$$\phi_{L_k}(\boldsymbol{x}) = I_k(\boldsymbol{x}) f_{k_0}(\boldsymbol{x}, f_{k_1}(\boldsymbol{x}, ...(f_{k_R}(\boldsymbol{x}, 1)))) \tag{11}$$

where each k_i is the i^{th} ancestor of k and

$$I_k(\boldsymbol{x}) = \Pi_{I \in Ancestors(k)} \frac{1}{2}(sign(\boldsymbol{\theta}_I^t \boldsymbol{x}) + (-1)^{C(I)}) \tag{12}$$

in the paper, best results are said to be obtained with

$$\phi_{L_k}(\boldsymbol{x}) = tanh(\sigma \boldsymbol{\theta}_k'^t \boldsymbol{x}) I_k(\boldsymbol{x}) \tag{13}$$

Algorithm

Primal for jointly learning Θ, Θ' and W

$$\min_{W,\Theta,\Theta'} P(W,\Theta,\Theta') = \frac{\lambda_W}{2} Tr(W^t W) + \frac{\lambda_\Theta}{2} Tr(\Theta^t \Theta) + \frac{\lambda_{\Theta'}}{2} Tr(\Theta'^t \Theta') + \sum_{i=1}^{N} L(y_i, \phi_L^t(\boldsymbol{x}_i) W^t \boldsymbol{x}_i)$$

where L is the hinge loss for binary classification

Algorithm

Primal stochiastic sub-gradient descent

W, Θ and Θ' updated as

$$W^{j+1} = W^j - \eta_j \nabla_W P(W^j, \Theta^j, \Theta'^j, \boldsymbol{x}_i)$$
$$\Theta^{j+1} = \Theta^j - \eta_j \nabla_\Theta P(W^j, \Theta^j, \Theta'^j, \boldsymbol{x}_i)$$
$$\Theta'^{j+1} = \Theta'^j - \eta_j \nabla'_\Theta P(W^j, \Theta^j, \Theta'^j, \boldsymbol{x}_i)$$

where η_j is the step size at iteration j

Algorithm

Primal stochiastic sub-gradient descent

and

$$\nabla_{\boldsymbol{w}_k} P(\boldsymbol{x}_i) = \lambda_W \boldsymbol{w}_k - \delta_i y_i \phi_{L_k}(\boldsymbol{x}_i)\boldsymbol{x}_i$$

$$\nabla_{\boldsymbol{\theta}_k} P(\boldsymbol{x}_i) = \lambda_\Theta \boldsymbol{\theta}_k - \delta_i y_i \sum_l tanh(\sigma \boldsymbol{\theta}_l'^t \boldsymbol{x}_i) \nabla_{\boldsymbol{\theta}_k} l_l(\boldsymbol{x}_i) \boldsymbol{w}_l^t \boldsymbol{x}_i$$

$$\nabla_{\boldsymbol{\theta}'_k} P(\boldsymbol{x}_i) = \lambda_\Theta \boldsymbol{\theta}'_k - \delta_i y_i \sigma (1 - tanh^2(\sigma \boldsymbol{\theta}_k'^t \boldsymbol{x}_i)) l_k(\boldsymbol{x}_i) \boldsymbol{w}_k^t \boldsymbol{x}_i \boldsymbol{x}_j$$

To make the optimisation tractable, and for ∇l to exist, we use a $tanh(.)$ parametrised by a scale parameter which is adaptively scaled to tend to sign(.) by the time convergence is reached

Performance comparison

Data Set	**Linear SVM**	**RBF-SVM**	**DC-Pred++**	**LDKL**
CovType Train=522,910 Test=58,102 Dims=54	A = 76.32%	A = 91.21% P = 131,785x	A = 95.19% P = 18.8x T = 372s	A = 88.21% P = 32x T = 4095s
Letter Train=12,000 Test=6,000 Dims=16	A = 73.02%	A = 97.20% P = 1548x	A = 95.90% P = 12.8x T = 1.2s	A = 96.30% P = 33x T = 243s

A = Accuracy(%), P = Prediction Time(times Linear SVM), T = Training Time(s)

Source: Jose et al., 2013 [1] and Hsieh et al., 2014 [2]

References I

[1] Cijo Jose, Prasoon Goyal, Parv Aggrwal, and Manik Varma.
Local deep kernel learning for efficient non-linear svm prediction.
In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 486–494, 2013.

[2] Cho-Jui Hsieh, Si Si, and Inderjit S Dhillon.
Fast prediction for large-scale kernel machines.
In *Advances in Neural Information Processing Systems*, pages 3689–3697, 2014.

[3] Christopher Williams and Matthias Seeger.
Using the nyström method to speed up kernel machines.
(EPFL-CONF-161322):682–688, 2001.

References II

[4] Petros Drineas and Michael W Mahoney.
On the nyström method for approximating a gram matrix for improved kernel-based learning.
The Journal of Machine Learning Research, 6:2153–2175, 2005.

[5] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar.
Ensemble nystrom method.
In *Advances in Neural Information Processing Systems*, pages 1060–1068, 2009.

[6] Cho-Jui Hsieh, Si Si, and Inderjit S Dhillon.
A divide-and-conquer solver for kernel support vector machines.
arXiv preprint arXiv:1311.0914, 2013.

References III

[7] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan.
Learning the kernel matrix with semidefinite programming.
The Journal of Machine Learning Research, 5:27–72, 2004.

[8] Asa Ben-Hur and William Stafford Noble.
Kernel methods for predicting protein–protein interactions.
Bioinformatics, 21(suppl 1):i38–i46, 2005.

[9] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Noble Grundy.
Gene functional classification from heterogeneous data.
In *Proceedings of the fifth annual international conference on Computational biology*, pages 249–255. ACM, 2001.

References IV

[10] Francis R Bach, Gert RG Lanckriet, and Michael I Jordan.
Multiple kernel learning, conic duality, and the smo algorithm.
In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 2004.

[11] Mehmet Gönen and Ethem Alpaydin.
Localized multiple kernel learning.
In *Proceedings of the 25th international conference on Machine learning*, pages 352–359. ACM, 2008.

[12] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet.
More efficiency in multiple kernel learning.
In *Proceedings of the 24th international conference on Machine learning*, pages 775–782. ACM, 2007.