rls_dual_mkl: A PFBS-based Implementation for Multiple Kernel Learning

(Jeremiah) Zhe Liu zhl112@mail.harvard.edu
Department of Biostatistics
Harvard School of Public Health
Boston, MA, 02115

Contents

1. Introduction

1.1. Multiple Kernel Learning Problem

Multiple kernel learning (MKL) (Bach et al. (2004)) is the process of finding an optimal kernel from a prescribed (convex) set $\mathcal{K}$ of basis kernels, for learning a real-valued function by regularization. In this work, we consider a RKHS $\mathcal{H} = \mathcal{H}_1 \oplus \mathcal{H}_2 \cdots \oplus \mathcal{H}_M$ with reproducting kernel $\mathbf{k} \in \mathcal{K} = \{\sum_{i=1}^M c_i \mathbf{k}_i | (c_i \geq 0 \forall i) \wedge \sum_{i=1} c_i = 1\}$ such that $f = \sum_{i=1}^M f_i, f_i \in \mathcal{H}_i$. By Micchelli and Pontil (2005), the problem of multiple kernel learning corresponds to find f^* such that:

$$\arg\min_{f \in \mathcal{H}} \Big\{ \frac{1}{n} \sum_{i=1}^n Q(\sum_{j=1}^M f_j(\mathbf{x}), \mathbf{y}) + \tau g\big((\sum_{j=1}^M ||f_j||_{\mathcal{H}})^2 \big) \Big\}$$

Rosasco et al. (2009) generalized above problem by taking Q to be square loss, $g(.) = \sqrt{.}$ and also impose L1 regularization, leading to the elastic-net-regulated problem:

$$\arg\min_{f \in \mathcal{H}} \Big\{ \frac{1}{n} \sum_{i=1}^n (\sum_{j=1}^M f_j(x_i) - y_i)^2 + \mu \sum_{j=1}^M ||f_j||_{\mathcal{H}}^2 + 2\tau \sum_{j=1}^M ||f_j||_{\mathcal{H}} \Big\} \tag{1}$$

1.2. Iterative PFBS Algorithm

By Theorem 1 of Rosasco et al. (2009), since the penalty function is lower semicontinuous, coercive, convex and one-homogenous, solution to problem 1 f^* is the unique fixed point of the the contractive mapping with step size σ:

$$\mathcal{T}_\sigma(f) = (\mathbf{I} - \pi_{\frac{\tau}{\sigma}K})\big(f - \frac{1}{2\sigma} \nabla_f [\frac{1}{n} ||f - y||^2]\big)$$

where $\pi_{\frac{\tau}{\sigma}K}(g)$ is a project operator which project g to $\mathcal{H}' = \{f \in \mathcal{H} \big| \ ||f_j||_{\mathcal{H}_j} \leq \frac{1}{\tau/\sigma} \ \forall j\}$, or more rigorously:

$$\pi_{\frac{\tau}{\sigma}K}(g) = \frac{\tau}{\sigma} v, \qquad \text{where } v = \arg \min_{v \in \mathcal{H}, ||v_j|| \leq 1} ||\frac{\tau}{\sigma} v - g||_{\mathcal{H}}^2$$

Above mapping can also be written in terms of Kernel matrices by generalizing representer theorem and write $f_j^*(x) = \sum_{i=1}^n \alpha_{ji}^T k_j(x_i, x) = \boldsymbol{\alpha}_j^T \mathbf{k}_j(x)$, where $\boldsymbol{\alpha}_j$ and $\mathbf{k}_j(x)$ are $n \times 1$ vectors. Further, if denote:

$$\begin{aligned}
\boldsymbol{\alpha}_{Mn\times 1} &= (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_M)^T \\
\mathbf{k}(x)_{Mn\times 1} &= (\mathbf{k}_1(x)^T, \ldots, \mathbf{k}_M(x)^T)^T \\
\mathbf{K}_{Mn\times Mn} &= \begin{bmatrix} \mathbf{K}_1 & \ldots & \mathbf{K}_M \\ \vdots & \ddots & \vdots \\ \mathbf{K}_1 & \ldots & \mathbf{K}_M \end{bmatrix}, \text{where } \mathbf{K}_i = \mathbf{k}_i(.)\mathbf{k}_i(.)^T \\
\mathbf{y}_{Mn\times 1} &= (y_{n\times 1}^T, \ldots, y_{n\times 1}^T)^T
\end{aligned}$$

The contraction mapping can be written as:

$$\mathcal{T}_\sigma(f) = (\mathbf{I} - \pi_{\frac{\tau}{\sigma}K})\big(\big[(1-\frac{\mu}{\sigma})\boldsymbol{\alpha} - \frac{1}{\sigma n}(\mathbf{K}\boldsymbol{\alpha} - \mathbf{y})\big]^T\mathbf{k}\big) \qquad \text{where} \tag{2}$$

$$\pi_{\frac{\tau}{\sigma}K}(g)_j = min\{1, \frac{||g_j||_{\mathcal{H}_j}}{\tau/\sigma}\} * \frac{g_j}{||g_j||_{\mathcal{H}_j}} = min\{1, \frac{\sqrt{\boldsymbol{\alpha}_j^T\mathbf{K}_j\boldsymbol{\alpha}_j}}{\tau/\sigma}\} * \frac{\boldsymbol{\alpha}_j^T\mathbf{k}_j}{\sqrt{\boldsymbol{\alpha}_j^T\mathbf{K}_j\boldsymbol{\alpha}_j}}$$

Thus the projection $\mathbf{I} - \pi_{\frac{\tau}{\sigma}K}$ corresponds to the soft-thresholding operator for $\boldsymbol{\alpha}_j$:

$$\mathbf{S}_{\frac{\tau}{\sigma}}(K, \boldsymbol{\alpha})_j = \frac{\boldsymbol{\alpha}_j^T}{\sqrt{\boldsymbol{\alpha}_j^T\mathbf{K}_j\boldsymbol{\alpha}_j}}(\sqrt{\boldsymbol{\alpha}_j^T\mathbf{K}_j\boldsymbol{\alpha}_j} - \frac{\tau}{\sigma})_+$$

Above discussions lead to below algorithm:

Algorithm 1: MKL Algorithm

set $\boldsymbol{\alpha}^0 = \mathbf{0}$

for $p = 1$ to `MAX_ITER` **do**

$\boldsymbol{\alpha}_0^p = (1 - \frac{\mu}{\sigma})\boldsymbol{\alpha}^{p-1} - \frac{1}{\sigma n}(\mathbf{K}\boldsymbol{\alpha}^{p-1} - \mathbf{y})$
$\boldsymbol{\alpha}^p = \mathbf{S}_{\frac{\tau}{\sigma}}(K, \boldsymbol{\alpha}_0^p)$

end for

return $f^{\texttt{MAX\_ITER}} = (\boldsymbol{\alpha}^{\texttt{MAX\_ITER}})^T\mathbf{k}$

1.3. Implementation Detail

1.3.1. Block-wise Update

Notice that in (2), $\mathcal{T}_\sigma$ updates $\boldsymbol{\alpha}$ by group, it is thus possible to write $\mathcal{T}$ at p^{th} step as:

$$\mathcal{T}_\sigma^p = [\mathcal{T}_{\sigma,1}^p, \mathcal{T}_{\sigma,2}^p, \ldots, \mathcal{T}_{\sigma,M}^p] \qquad \text{with} \qquad \mathcal{T}_{\sigma,j}^p = \mathbf{S}_{\frac{\tau}{\sigma}}\Big(K, \boldsymbol{\alpha}_0\Big)_j$$

$$\boldsymbol{\alpha}_0 = (1 - \frac{\mu}{\sigma})\boldsymbol{\alpha}_j^{p-1} - \frac{1}{n\sigma} * \boldsymbol{\epsilon}^{p-1} \qquad \text{where } \boldsymbol{\epsilon}^{p-1} = (\sum_{j=1}^{M}\mathbf{K}_j\boldsymbol{\alpha}_j^{p-1} - y)$$

by using above method we are able to avoid working directly with the $Mn \times Mn$ matrix $\mathbf{K}$ (as defined earlier in section 1.2), which led to reduced memory cost [1] and reduced difficulty in selecting stepsize and regularization parameters.

1. $O(Mn^2)$ instead of $O(M^2n^2)$

1.3.2. Choice of Stepsize

Based on Bach et al. (2004), it can be shown that a suitable choice of σ is $\sigma = \frac{1}{4}(a * L_{min} + b * L_{max}) + \mu$, where (b, a) denotes the lower/upper bound on the eigenvalue of $\mathbf{K}$, and (L_{min}, L_{max}) denotes the lower/upper bound of $\nabla^2 Q(\mathbf{f}, \mathbf{y})$.

In the context where Q is square loss (i.e. $\nabla^2_{\mathbf{f}} Q(\mathbf{f}, \mathbf{y}) = 2$), we have:

$$\sigma = \frac{1}{2}(a + b) + \mu$$

A naive choice of a would be the largest eigenvalue of $\mathbf{K}_{nM\times nM}$, which not only is computationally expensive but also leads to overly slow convergence. In pactice, if denote the maximum eigenvalue of each kernel matrix K_j to be a_j, it is found that setting a to be $\max\limits_{j\in\{1,..,M\}}(a_j)$ is suffice to guarantee convergence. This is because the mapping $\boldsymbol{\alpha}^{p-1} \mapsto \boldsymbol{\alpha}_0^p$ can be written as:

$$\boldsymbol{\alpha}_0 = (1 - \frac{\mu}{\sigma})\boldsymbol{\alpha}_j^{p-1} - \frac{1}{n} * \sum_{j=1}^{M} \frac{1}{\sigma}(\mathbf{K}_j \boldsymbol{\alpha}_j^{p-1} - \frac{y}{n})$$

As shown, in $\boldsymbol{\alpha}^{p-1} \mapsto \boldsymbol{\alpha}_0^p$ we actually updated $\boldsymbol{\alpha}^{p-1}$ M times, with step size $\frac{1}{\sigma}(\mathbf{K}_j\boldsymbol{\alpha}_j^{p-1} - \frac{y}{n})$ in each step. It is thus sufficient to find a a that properly scale the magnitude of all $||\mathbf{K}||$, leading natually to the choice $a = \max\limits_{j\in\{1,..,M\}}(a_j)$.

1.3.3. Effect of μ and τ

The convergence of the aforementioned procedure is guaranteed by Banach Fixed Point theorem, given proper choice of σ.

Zou and Hastie (2005)

Rosasco et al. (2009)

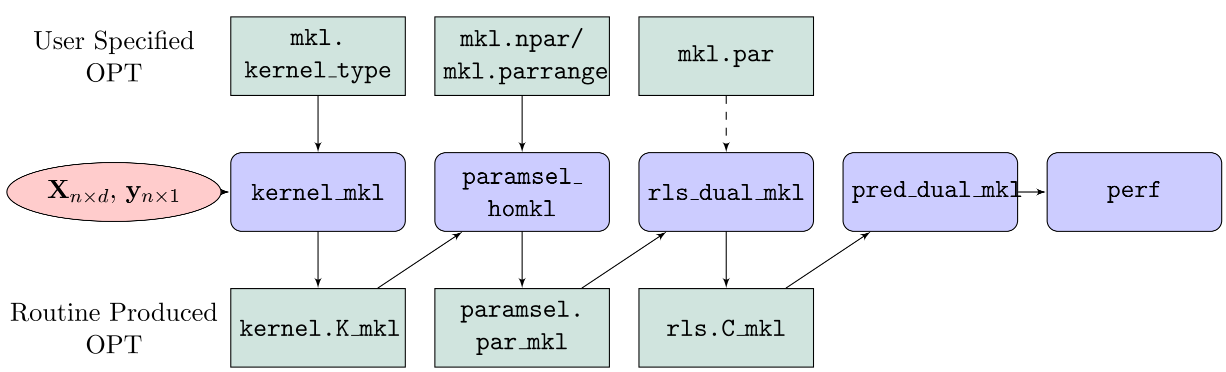
Contraction mapping

$||(1 - \frac{\mu}{\sigma})\mathbf{I} - \frac{1}{\sigma n}\mathbf{K}||$

Continuation strategy.

$\frac{\tau}{\sigma}_{max} = \frac{||y||^2}{||\mathbf{K}||}$

2. Software Structure and Usage



1. Parameter Specification
   - util/gurls_defopt_mkl
2. Kernel Generation
   - kernel/kernel_mkl
3. Parameter Selection
   - kernel/paramsel_homkl
   - util/paramsel_L1ratioguesses
4. Optimization
   - optimizers/rls_dual_mkl
   - optimizers/rls_dual_mkl_pfbs
   - util/ConsoleProgressBar
   - summary/plot_mkl_L1path
5. Prediction
   - kernel/predkernel_traintest
   - optimizers/pred_dual_mkl
6. Performance assessment
   - perf/perf_rmsestd

3. Example

[2]

2. LIBSVM Data Repository: 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html'

References

F Bach, G Lanckriet, and M Jordan. Multiple kernel learning, conic duality, and the smo algorithm. *ACM International Conference Proceeding Series*, 69, 2004.

C Micchelli and M Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:10991125, 2005.

L Rosasco, S Mosci, M Santoro, A Verri, and S Villa. Iterative projection methods for structured sparsity regularization. Technical report, MIT, 2009.

H Zou and T Hastie. Regularization and variable selection via the elastic net. *J. R. Statist. Soc. B*, 67, Part 2:301320, 2005.