Efficient Multiple Kernel Learning (2)

Lei Tang

Arizona State University

May. 29th, 2007

For simplicity, we assume all Kernel matrix have the same trace (without scaling problem). Given $G_1, G_2, \cdots, G_p$, the weight for each kernel is β_k, and $\sum \beta_k = 1$.

Goal

$$\begin{aligned} \min_{\beta} \max_{\alpha} \quad & \alpha^T e - \frac{1}{2}\alpha^T \mathrm{diag}(y) \left(\sum_{k=1}^{p} \beta_k G_k \right) \mathrm{diag}(y) \alpha \\ s.t. \quad & \alpha^T y = 0 \\ & C \geq \alpha \geq 0 \\ & \beta^T e = 1, \beta \geq 0 \end{aligned}$$

$$
\begin{aligned}
& \min_{\beta} \max_{\alpha} \alpha^T e - \frac{1}{2}\alpha^T \mathrm{diag}(y) \left(\sum_{k=1}^{p} \beta_k G_k \right) \mathrm{diag}(y)\alpha \\
= \quad & \max_{\alpha} \min_{\beta} \alpha^T e - \frac{1}{2}\alpha^T \mathrm{diag}(y) \left(\sum_{k=1}^{p} \beta_k G_k \right) \mathrm{diag}(y)\alpha \\
= \quad & \max_{\alpha} \alpha^T e - \max_{sum_k \beta_k = 1} \frac{1}{2} \sum_{k=1}^{p} \beta_k \left(\alpha^T \mathrm{diag}(y) \; G_k \; \mathrm{diag}(y)\alpha \right) \\
= \quad & \max_{\alpha} \alpha^T e - \max_{i} \left(\frac{1}{2}\alpha^T \mathrm{diag}(y) \; G_k \; \mathrm{diag}(y)\alpha \right)
\end{aligned}
$$

QCQP

So the problem can be reformulated as

$$
\begin{aligned}
max_{alpha} \quad & \alpha^T e - \frac{1}{2} t \\
s.t \quad & t \geq \alpha^T y = 0 \\
& C \geq \alpha \geq 0 \\
& t \geq \frac{1}{2} \alpha^T \mathrm{diag}(y)\ G_k\ \mathrm{diag}(y) \alpha \quad \forall k
\end{aligned}
$$

This is a QCQP problem, which can be solved by general optimization package. But it does not scale up!!

Let

$$S_k(\alpha) = \alpha^T e - \frac{1}{2}\alpha^T \mathrm{diag}(y) G_k \mathrm{diag}(y)\alpha$$

As $\sum \beta_k = 1$, the objective becomes

$$\min_{\beta} \max_{\alpha} \sum_{k=1}^{p} \beta_k S_k(\alpha)$$

So the goal is equivalent to

$$\begin{aligned}
\min_{\beta} \max_{\alpha} \quad & \sum_{k=1}^{p} \beta_k S_k(\alpha) \\
s.t. \quad & \alpha^T y = 0 \\
& C \geq \alpha \geq 0 \\
& \beta^T e = 1, \beta \geq 0
\end{aligned}$$

$$
\begin{aligned}
\min_{\beta} \max_{\alpha} \quad & \sum_{k=1}^{p} \beta_k S_k(\alpha) \\
s.t. \quad & \alpha^T y = 0 \\
& C \geq \alpha \geq 0 \\
& \beta^T e = 1, \beta \geq 0
\end{aligned}
$$

Assume α^* is the optimal, let $\theta^* := \sum_{k=1}^{p} \beta_k S_k(\alpha)$ would be the maximal,

$$
\sum_{k=1}^{p} \beta_k S_k(\alpha) \leq \theta^*
$$

$$\begin{aligned} \mathit{min} \quad & \theta \\ s.t. \quad & \beta \geq 0, \sum_k \beta_k = 1 \\ & \sum_{k=1}^{p} \beta_k S_k(\alpha) \leq \theta \\ & \text{for all } \ \alpha \in \mathcal{R}^N, 0 \leq \alpha \leq C, \alpha^T y = 0 \end{aligned}$$

Note that θ and β are linearly constrained with infinitely many constraints for all possible α.

Algorithms to solve SILP

$$\begin{array}{ll} max & \theta \\ s.t. & \beta \geq 0, \sum_k \beta_k = 1, \sum_{k=1}^{p} \beta_k S_k(\alpha) \geq \theta \\ & \text{for all } \ \alpha \in \mathcal{C} \end{array}$$

Column Generation

1. *Restricted master problem:* Compute optimal (β, θ) for a restricted subsets of constraints (Typical Linear Programming problem)
2. Add the constraints that maximize the constraint violation for the given intermediate solution (β, θ). (Exactly an SVM dual formulation with a combined kernel)

Idea is exactly the same as *cutting plane*

Algorithms to solve SILP

$$\begin{aligned} &\max && \theta \\ &s.t. && \beta \geq 0, \sum_k \beta_k = 1, \sum_{k=1}^{p} \beta_k S_k(\alpha) \geq \theta \\ & && \text{for all} \quad \alpha \in \mathcal{C} \end{aligned}$$

Column Generation

1. *Restricted master problem:* Compute optimal (β, θ) for a restricted subsets of constraints (Typical Linear Programming problem)
2. Add the constraints that maximize the constraint violation for the given intermediate solution (β, θ). (Exactly an SVM dual formulation with a combined kernel)

Idea is exactly the same as *cutting plane*

Other tricks to accelarate calculation

- Step 2 is not necessary to be exact if β is still far away from optimal.
- *Chunking*: maintain a smaller number of optimization variables.
- *Warm Start* for step 1 to solve LP
- Specific data structure to maintain the gram matrix (tries), save memory. Kernel caching.
- Parallel processing

Works up to 20 kernels and one million examples

Another formulation

Goal

$$\begin{aligned}
\min_{\beta} \max_{\alpha} \quad & \alpha^T e - \frac{1}{2}\alpha^T \mathrm{diag}(y) \left(\sum_{k=1}^{p} \beta_k G_k \right) \mathrm{diag}(y)\alpha \\
s.t. \quad & \alpha^T y = 0 \\
& C \geq \alpha \geq 0 \\
& \beta^T e = 1, \beta \geq 0
\end{aligned}$$

Let

$$J(\beta) = \begin{cases} \max_{\alpha} & \alpha^T e - \frac{1}{2}\alpha^T \mathrm{diag}(y) \left(\sum_{k=1}^{p} \beta_k G_k \right) \mathrm{diag}(y)\alpha \\ \quad s.t. & \alpha^T y = 0 \\ & C \geq \alpha \geq 0 \end{cases}$$

The problem can be reformulated as

$$\min_{\beta} J(\beta) \text{ such that } \sum_{k=1}^{p} \beta_k = 1, \beta \geq 0$$

Let α^* be the optimal value for $J(\beta)$, then

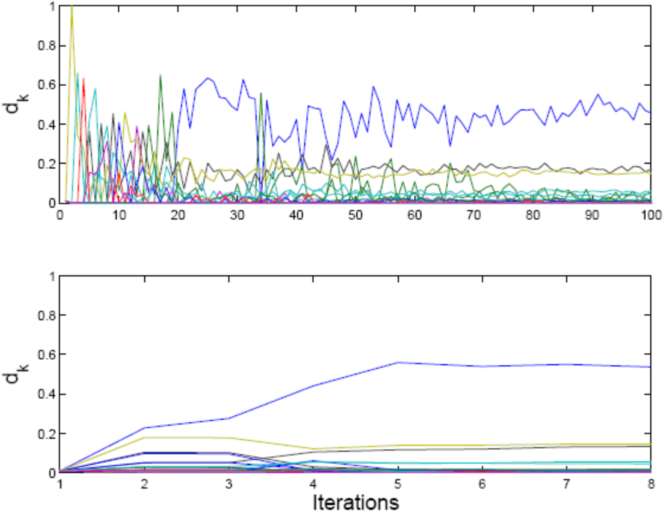
$$\begin{aligned} J(\beta) &= {\alpha^*}^T e - \frac{1}{2}{\alpha^*}^T \mathrm{diag}(y) \left(\sum_{k=1}^{p} \beta_k G_k\right) \mathrm{diag}(y)\alpha^* \\ \frac{\partial J(\beta)}{\partial \beta_k} &= -\frac{1}{2}{\alpha^*}^T \mathrm{diag}(y)\ G_k\ \mathrm{diag}(y)\alpha^* \end{aligned}$$

Steepest Decent Algorithm

- For fixed β, calculate the optimal $J(\beta)$; (Exactly an SVM problem)
- Calcuate a decent direction for β such that the constraint is satisfied.
- Update *beta* by finding the step length using some line search method (e.g. Armijo's rule). This involves multiple evaluatons of $J(\beta)$ (A single kernel SVM) with some small variants of β. This could be speed up by initilizing the SVM with α^*.

Cuttting Plane v.s. Steepest Decent

- Cutting plane graduatelly add constraints whereas steepest decent involves fixed number of constraints in each iteration.
- Both involves calculation of SVM in each iteration. Steepest Decent involves a little more calculation while computing the step length.
- Convergence Analysis: Both converge. But Cutting planes method are unstable, especially when the number of lower-bounding affine functions is small.



1
0.8
0.6
0.4
0.2
0
d_k
0
10
20
30
40
50
60
70
80
90
100
1
0.8
0.6
0.4
0.2
0
d_k
1
2
3
4
5
6
7
8
Iterations

Credit $M = 208$

Algorithm	# Kernel	Accuracy	Time (s)
SILP	10.4 ± 1.9	86.4 ± 1.4	124 ± 28
Our	16.8 ± 5.4	86.3 ± 1.4	32.7 ± 9.7

Ionosphere $M = 442$

Algorithm	# Kernel	Accuracy	Time (s)
SILP	19.9 ± 2.1	92.2 ± 1.5	152 ± 39
Our	30.5 ± 7.0	92.3 ± 1.4	18.1 ± 5.8

Sonar $M = 793$

Algorithm	# Kernel	Accuracy	Time (s)
SILP	29.1 ± 3.5	79.2 ± 4.6	383 ± 113
Our	46.0 ± 7.6	78.6 ± 4.2	21.8 ± 16.9

Referneces

1. Large Scale Multiple Kernel Learning, JMLR, 2006
2. More Efficiency in Multiple Kernel Learning, ICML, 2007