Large-scale Multiple Kernel and Multitask Learning

Marius Kloft
Lille, July 11, 2015

Outline

1 Multiple Kernel Learning (MKL)

2 (Kernel-based) Multi-task Learning

3 Multi-task Multiple Kernel Learning (MT-MKL)
Multiple Kernel Learning (MKL)

Problem Setting

Given:
- Observed data \((x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}\)
- Kernels \(k_1, \ldots, k_M\) corresponding to feature maps \(\phi_1, \ldots, \phi_M\)

Aim:
- Find good combination of kernels \(k_\beta := \sum_{m=1}^{M} \beta_m k_m, \beta_m \geq 0\)
- Corresponds to finding a good weighting of feature spaces

\[
\phi_\beta := \begin{pmatrix}
\sqrt{\beta_1} \phi_1 \\
\vdots \\
\sqrt{\beta_M} \phi_M
\end{pmatrix}
\]

as \(\langle \phi_\beta(x), \phi_\beta(\tilde{x}) \rangle = \sum_{m=1}^{M} \langle \sqrt{\beta_m} \phi_\beta(x), \sqrt{\beta_m} \phi_\beta(\tilde{x}) \rangle = \sum_{m=1}^{M} \beta_m k_m(x, \tilde{x})\).

Chapelle et al. (2002); Lanckriet et al. (2004); Bach et al. (2004)
Multiple Kernel Learning (MKL)

Problem:
▶ Grid search of optimal weights not efficient

MKL core idea:
▶ Many learning machines formulated as optimization problems (e.g., SVM)
▶ Optimize also over kernel weights!

(Lanckriet et al., 2004; Bach et al., 2004)

\[(\ell_p\text{-norm}) \text{ Multiple Kernel Learning}\]

\[
\min_{\beta, w, b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} l\left(y_i \sum_{m=1}^{M} \langle w_m, \sqrt{\beta_m} \phi_m(x_i) \rangle + b \right) \\
\text{s.t. } \|\beta\|_p \leq \lambda \quad \text{w.l.o.g. } 1, \quad \beta \geq 0, \quad p \in [1, \infty]
\]

(Kloft et al., 2009, 2011)

Geometry of \(\ell_p\)-norms

▶ \(\|\beta\|_p := \sqrt[p]{\sum_{m=1}^{M} |\beta_m|^p}\)

<table>
<thead>
<tr>
<th>(p)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p = 1)</td>
<td>classic MKL</td>
</tr>
<tr>
<td>(p \in [1, \infty])</td>
<td>(\ell_p)-norm MKL</td>
</tr>
<tr>
<td>(p = \infty)</td>
<td>SVM</td>
</tr>
</tbody>
</table>

Best results usually achieved by optimizing over \(p \in [1, \infty]\).
Convexity of MKL

Initially MKL is not convex:

\[
\min_{\beta, w, b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} l\left(y_i \sum_{m=1}^{M} \left< w_m, \sqrt{\beta_m} \phi_m(x_i) \right> + b \right) \\
\text{s.t. } \|\beta\|_p \leq 1, \quad \beta \geq 0
\]

But, by variable substitution \(w^\text{new}_m := \sqrt{\beta_m} w^\text{old}_m\), equivalently

\[
\min_{\beta, w, b} \frac{1}{2} \sum_{m=1}^{M} \frac{\|w_m\|^2}{\beta_m} + C \sum_{i=1}^{n} l\left(y_i \sum_{m=1}^{M} \left< w_m, \phi_m(x_i) \right> + b \right) \\
\text{s.t. } \|\beta\|_p \leq 1, \quad \beta \geq 0
\]

which is convex.

How to optimize?

Standard Optimization Approach (1)

Alternate between optimizing w.r.t. \(\beta\) and optimizing w.r.t. to \(w\) and \(b\).

E.g., block coordinate descent:

1. For fixed \(\beta\), optimizing w.r.t. \(w\) and \(b\) is just carried out by standard SVM solver.
2. For fixed \(w\) and \(b\), optimizing w.r.t. \(\beta\) can be done in closed form:

\[
\min_{\beta} \frac{1}{2} \sum_{m=1}^{M} \frac{\|w_m\|^2}{\beta_m} + \lambda \|\beta\|^p_p \text{ has optimum for } \beta_m \propto \frac{p+1}{\sqrt{\|w_m\|^2}}.
\]
Standard Optimization Approach (2)

**Block Coordinate Descend Algorithm**

1: initialize kernel weights $\beta = (\beta_1, \ldots, \beta_M)$ (e.g., uniformly)
2: repeat
3: train SVM($\sum_{m=1}^{M} \beta_m k_m$)
4: compute $\|w_m\|^2$ for all $m = 1, \ldots, M$
5: $\beta_m := p + 1 \sqrt{\|w_m\|^2}$ for all $m = 1, \ldots, M$
6: $\beta := \beta / \|\beta\|^p$
7: until converged (relative duality gap $< \epsilon$ e.g. 0.01 or 0.001)

Note that SVM may be trained in the dual, if required:

**Representer Theorem**

Given fixed kernel weights, in the optimum:

$$\forall m = 1, \ldots, M : w_m = \sum_{i=1}^{n} \alpha_i \beta_m \varphi_m(x_i)$$

Can we do better?

**Interleaved Optimization**

Core idea:

- directly code the update of kernel weights into SVM solver
  (Sonnenburg et al., 2006; Kloft et al., 2011)
- such a solver based on SVMLight is part of the SHOGUN toolbox
  (Sonnenburg et al., 2010)

Features:

- interfaces to Python, MATLAB, R, Octave, Java, C++, C#, Lua
Results

$M = 50$ kernels, $d = 784$ features

$n = 1000$, $d = 784$ features

Kloft et al. (2011)
Another Fast Solver

(Vishwanathan, Sun, Ampornpunt, and Varma, 2010)

Idea:

- Remove dependency on $\beta$,

$$
\max_{\alpha : y^T \alpha = 0, 0 \leq \alpha \leq C} \quad \min_{\beta} \quad 1^T \alpha - \frac{1}{2} \left( (\alpha \circ y)^T \sum_{m=1}^{M} \beta_m K_m(\alpha \circ y) \right) \\
\text{s.t.} \quad \|\beta\|_p \leq 1, \quad \beta \geq 0
$$

- Sequential minimal optimization (SMO)
  - optimize one $(\alpha_i, \alpha_j)$ pair at a time, use shrinking
  - Implementation based on LIBSVM

Results

(Vishwanathan, Sun, Ampornpunt, and Varma, 2010)
Extensiosn

The discussed methods can be extended to other setups including

- Regression (Cortes et al., 2009)
- Multi-class Classification (Zien and Ong, 2007)
- One-class Classification (Sonnenburg et al., 2006)
- Localized MKL (LMKL) (Gönen and Alpaydin, 2008)
- Mixed-norm based MKL (Nath et al., 2009)
- ...

Convex Localized MKL

(Lei, Binder, Dogan, and Kloft, 2015)

Need of flexible kernel weights varying over input space

- **Clustering**: partition the data into clusters \( c = 1, \ldots, C \)
  
  \( p_c(x) := \text{likelihood of } x \text{ falling into cluster } c \)

- **Idea**: each cluster \( c \) gets its own kernel weights
  \( \beta_c = (\beta_{c1}, \ldots, \beta_{cM}) \geq 0 \quad \text{s.t. } \|\beta_c\|_p \leq 1 \)

- Alternating optimization:
  1. SVM(\( K_\beta \)),
     \[ k_\beta(x_i, x_j) := \sum_{m=1}^{M} \sum_{c=1}^{C} p_c(x_i)p_c(x_j)\beta_{cm}k_m(x_i, x_j) \]
     \[ =: \hat{\beta}_m(x_i, x_j) \]
  2. Analytical updates \( \beta_{cm} \propto p_c^{n+1} \left\| w_{cm} \right\|^2 \)
  3. Representer theorem: \( w_{cm} = \sum_{i=1}^{n} \alpha_i p_c(x_i)\beta_{cm}\phi_m(x_i) \)
Discussion

Limitations:
- on-the-fly kernel computation too slow for large number of training points

No parallelization!

Potential directions:
- Primal solver (LIBLINEAR, SGD, ...)
  - Feature hashing
  - Random features for RBF kernels (similar tricks also for other kernels?)
- Efficient kernel computation (e.g., string kernels)

Does it really work?

Application: Transcription Start Site (TSS) Detection

- 5 heterogeneous Kernels
  - based on sequence alignment, nucleotide / n-gram frequencies (up-/downstream), secondary properties (binding, stacking energies).
- baseline SVM = winner of international comparison of 19 models (Abeel et al., 2009)
(Kernel-based) Multi-task Learning

Problem setting

- Model quality often limited by insufficient training data.

Multi-task Learning (MTL)

- Using data or information from multiple learning tasks that have similar characteristics
- in order to mutually improve their prediction accuracies.

Example

In Biology, tasks can correspond to organisms.

- Organisms share evolutionary history
- Many biological processes are conserved between organisms.
Graph-based MTL

Not all tasks are equally similar
⇒ use task similarity matrix
\[ A = (A_{st})_{1 \leq s, t \leq T} \]

Graph-based multi-task learning (Evgeniou et al., 2005)

Promote similar weights for similar tasks,
\[
\text{Regularizer} = \sum_{s,t=1}^{T} A_{st} \| w_s - w_t \|^2 = \sum_{s,t=1}^{T} L_{st} w_s^T w_t. \]

where \( w_s \) is the weight vector of the \( s \)th task and \( L = D - A \) is the graph Laplacian of \( A \).

\[ D = (D_{st}), \quad D_{st} := \delta_{st} \sum_u A_{su}, \text{ is the degree matrix of } A. \]

Graph-based MTL (2)

Graph-based multi-task learning (Evgeniou et al., 2005)

Given a convex loss function \( \ell \),
\[
\min_{w_1, \ldots, w_T \in \mathbb{R}^d} \frac{\lambda}{2} \sum_{t=1}^{T} \| w_t \|^2 + \frac{1}{2} \sum_{s,t=1}^{T} L_{st} w_s^T w_t + C \sum_{i=1}^{n} \ell \left( y_i \langle w_t(i), x_i \rangle \right) \]

where \( t(i) \) indexes the task corresponding to the \( i \)th example.

Can be more compactly written as:
\[
\min_{W=(w_1, \ldots, w_T) \in \mathbb{R}^{d \times T}} \frac{1}{2} \text{tr} \left( W (\lambda I + L) W^T \right) + C \sum_{i=1}^{n} \ell \left( y_i \langle w_t(i), x_i \rangle \right) \]
Dualization

Can show:

\[
\text{Dual} \quad \max_{\alpha : y^\top \alpha = 0, 0 \leq \alpha \leq C} \quad 1^\top \alpha - \frac{1}{2} (\alpha \circ y)^\top \tilde{K} (\alpha \circ y)
\]

where for given \( \Sigma = (\Sigma_{st})_{s,t=1}^T \), the \textbf{multitask kernel matrix} \( \tilde{K} \) is defined as \( \tilde{k}_{ij} := \Sigma_{t(i),t(j)} k_{ij} \).

Can use any dual SVM solver (LIBSVM, SVMLight, ...)

Can we do better?

Solver: Dual Coordinate Ascend

Idea:

- Optimize w.r.t. a single training example at a time \,(Hsieh, Chang, Lin, Keerthi, and Sundararajan, 2008)\n
\[
\arg\max_d \quad d - \frac{1}{2} d^2 \Sigma_{t(i),t(j)} \langle x_i, x_i \rangle - d y_i \langle w_{t(i)}, x_i \rangle
\]

- where \( w_s = \sum_{i=1}^n \alpha_i y_i \Sigma_{s,t(i)} x_i \).

- Optimum:

\[
d = \frac{1 - y_i \langle w_{t(i)}, x_i \rangle}{\Sigma_{t(i),t(i)} \langle x_i, x_i \rangle}.
\]
MTL training algorithm (Widmer, Kloft, Görnitz, and Rätsch, 2012)

1: **init** $\alpha := 1$ and $w_t := 0$ for all $t = 1, \ldots, T$
2: **repeat**
3: for $i = 1, \ldots, n$ do
4: compute step size $d = \frac{1 - y_i \langle w_t(i), x_i \rangle}{\sum_{t(i), t(i)} \langle x_i, x_i \rangle}$
5: project onto constraints, $d := \max(0, \min(C, \alpha_i + d)) - \alpha_i$
6: for $s = 1, \ldots, T$ do
7: $w_s := w_s + \sum_{s, t(i)} dy_i x_i$
8: end for
9: end for
10: **until** converged (relative duality gap $< \epsilon := 0.01$ or 0.001)

---

Computational Experiments

Data sets:

<table>
<thead>
<tr>
<th>Data set</th>
<th>#dim</th>
<th>#examples</th>
<th>#tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gauss2D</td>
<td>2</td>
<td>$1 \cdot 10^5$</td>
<td>2</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>44</td>
<td>474</td>
<td>3</td>
</tr>
<tr>
<td>MNIST-MTL</td>
<td>784</td>
<td>$9.0 \cdot 10^3$</td>
<td>3</td>
</tr>
<tr>
<td>Land Mine</td>
<td>9</td>
<td>$1.5 \cdot 10^4$</td>
<td>29</td>
</tr>
<tr>
<td><strong>Splicing</strong></td>
<td>$6 \cdot 10^6$</td>
<td>$6.4 \cdot 10^6$</td>
<td>4</td>
</tr>
</tbody>
</table>
Results: Convergence

(a) Gauss2D

(b) Breast cancer

(c) MNIST-MTL

(d) Land Mine

Results: Large-scale SHOGUN Splice Experiment

- Uses COFFIN framework by Sonnenburg and Franc (2010)
- on-the-fly computation of weighted-degree string kernels
- hashing of $k$-grams to dense lower dimensional vectors

$d = 6 \cdot 10^6$

$T = 4$
Limitations

What if the task similarity matrix is not given?

Multi-task Multiple Kernel Learning (MT-MKL)
Learning the Task Similarities: **Approach 1**
(Blanchard, Lee, and Scott, 2011)

Estimate task similarity matrix $A$

$$A_{st} := \frac{1}{|I_s||I_t|} \sum_{i \in I_s, j \in I_t} k(x_i, x_j)$$

where $I_s, I_t \subset \{1, \ldots, n\}$ denote the indices corresponding to the $s$th and $t$th task, respectively.

Learning the Task Similarities: **Approach 2**
(Argyriou, Evgeniou, and Pontil, 2008)

Optimize over task similarities

$$\min_{W=(w_1, \ldots, w_T) \in \mathbb{R}^{d \times T}, \Sigma \in \mathbb{R}^{T \times T}} \frac{1}{2} \text{tr} \left( W \Sigma^{-1} W^\top \right) + C \sum_{i=1}^n \ell \left( y_i \langle w_{t(i)}, x_i \rangle \right),$$

s.t. $\Sigma \succ 0$, $\text{tr} \left( \Sigma \right) = 1$

- Optimum attained for $\Sigma = \frac{\sqrt{W^\top W}}{\text{tr}(\sqrt{W^\top W})}$.
- Hence equivalent to:

$$\min_{W=(w_1, \ldots, w_T) \in \mathbb{R}^{d \times T}} \frac{1}{2} \text{tr} \left( \sqrt{W^\top W} \right) + C \sum_{i=1}^n \ell \left( y_i \langle w_{t(i)}, x_i \rangle \right).$$

nuclear norm
Learning the Task Similarities: **Approach 3**

(Widmer, Kloft, Sreedharan, and Rätsch, 2015)

**Idea:**
- Given candidate task similarity matrices $\Sigma_1, \ldots, \Sigma_M$
- Learn a good mixture $\Sigma_\beta := \sum_{m=1}^{M} \beta_m \Sigma_m$

**Multi-task multiple kernel learning**

\[
\min_{W_1, \ldots, W_M, \beta: ||\beta||_p \leq 1} \frac{1}{2} \sum_{m=1}^{M} \operatorname{tr} \left( W_m \Sigma_m^{-1} W_m^\top \right) \beta_m + C \sum_{i=1}^{n} \ell \left( y_i \sum_{m=1}^{M} \langle w_{mt(i)}, \phi_m(x_i) \rangle \right)
\]

**Dual**

\[
\max_{\alpha: y^\top \alpha = 0, 0 \leq \alpha \leq C} \min_{\beta: ||\beta||_p \leq 1} 1^\top \alpha - \frac{1}{2} (\alpha \circ y)^\top \left( \sum_{m=1}^{M} \beta_m \tilde{K}^{(m)} \right) (\alpha \circ y)
\]

where $\tilde{K}^{(m)}$ is defined as $\tilde{k}_{ij}^{(m)} := \sum_{t(i), t(j)} k_{ij}$.

---

**Hierarchical Decomposition**

- **Taxonomy** $G$ gives us reasonable groups
  - Idea is to refine structure
- Meta-tasks are defined by taxonomy $G$

\[
\mathcal{I}_G = \{ \text{leaves}(\text{node}) | \text{node} \in G \}
\]

---

**Does it really work?**
Application: Transcription Start Site (TSS) Detection
(Widmer, Kloft, Sreedharan, and Rätsch, 2015)

\[ T = 9 \text{ organisms/tasks} \]
\[ n = 21,600 \text{ training instances} \]
\[ \text{Weighted-degree kernel using a window of 1200 nucleotides around candidate sites.} \]

MT-MKL help in average by about 2.5%. AUC

Conclusion

- **Multiple Kernel Learning (MKL)**
  1. Block coordinate descent (SHOGUN: interleaved \( \beta \) updates hacked into SVMLight)
  2. SMO on fully dualized objective

- **Multi-task learning (MTL)**
  1. Standard SVM dual with multi-task kernel (MTK)

  How to learn the task similarity matrix?

- **Multi-task multiple kernel learning (MT-MKL)**

For all the above: LIBLINEAR-style dual coordinate ascent with interleaved MKL updates.

Still a lot to be done to get this much faster...
Limitations and Discussion

No parallelization.

Directions:

- Speed-up dual coordinate ascent
  - distributed (Lee and Roth, 2015)
  - randomized (e.g., Csiba, Qu, and Richtárik, 2015)
- Stochastic gradient descent
- Non-linear kernels, random features, …

References I


References II


