Output Kernel Learning Methods

Francesco Dinuzzo
MPI for Intelligent Systems
Tübingen, Germany
fdinuzzo@tuebingen.mpg.de

Cheng Soon Ong
NICTA,
Melbourne, Australia
chengsoon.ong@unimelb.edu.au

Kenji Fukumizu
Institute of Statistical Mathematics,
Tachikawa, Tokyo, Japan
fukumizu@ism.ac.jp

Abstract: A rather flexible approach to multi-task learning consists in solving a regularization problem where a suitable kernel is used to model joint relationships between both inputs and tasks. Since specifying an appropriate multi-task kernel in advance is not always possible, estimating one from the data is often desirable. Herein, we overview a class of techniques for learning a multi-task kernel that can be decomposed as the product of a kernel on the inputs and one on the task indices. The kernel on the task indices (output kernel) is optimized simultaneously with the predictive function by solving a joint two-level regularization problem.

Keywords: regularization, multi-task learning, kernel learning

1 Learning Multi-Task Kernels

Predictive performances of kernel-based regularization methods are highly influenced by the choice of the kernel function. Such influence is especially evident in the case of multi-task learning where, in addition to specifying input similarities, it is crucial to correctly model inter-task relationships. Designing the kernel allows to incorporate domain knowledge by properly constraining the function class over which the solution is searched. Unfortunately, in many problems the available knowledge is not sufficient to uniquely determine a good kernel in advance, making it highly desirable to have data-driven automatic selection tools. This need has motivated a fruitful research stream which has led to the development of a variety of techniques for learning the kernel.

For a broad class of multi-task (or multi-output) learning problems, a kernel can be used to specify the joint relationships between inputs and tasks [1]. Generally, it is necessary to specify similarities of the form $K((x_1, i), (x_2, j))$ for every pair of input data (x_1, x_2) and every pair of task indices (i, j). However, a very common way to simplify modeling is to utilize a multiplicative decomposition of the form

$$K((x_1, i), (x_2, j)) = K_X(x_1, x_2)K_Y(i, j),$$

where the *input kernel* K_X is decoupled from the *output kernel* K_Y. The same structure can be equivalently represented in terms of a matrix-valued kernel

$$H(x_1, x_2) = K_X(x_1, x_2) \cdot \mathbf{L}, \tag{1}$$

where $\mathbf{L}$ is a symmetric and positive semidefinite matrix with entries $\mathbf{L}_{ij} = K_Y(i, j)$.

Even after imposing such simplified model, specifying the inter-task similarities in advance may still be impractical. Indeed, it is often the case that multiple learning tasks are known to be related, but no precise information about the structure or the intensity of such relationships is available. Simply fixing $\mathbf{L}$ to the identity is clearly suboptimal since it amounts to share no information between the tasks. On the other hand, wrongly specifying the entries may lead to a severe performance degradation. It is therefore clear that, whenever the task relationships are subject to uncertainty, learning them from the data is the only meaningful way to proceed.

The most widely developed approach to automatic kernel selection, known as Multiple Kernel Learning (MKL), consists in learning a conic combination of basis kernels of the form

$$K = \sum_{k=1}^{N} d_k K_k, \qquad d_k \geq 0.$$

Appealing properties of MKL methods include the ability to perform selection of a subset of kernels via sparsity, and tractability of the associated optimization

problem, typically (re)formulated as a convex program. Apparently, the MKL approach can be also used to learn a multi-task kernel of the form

$$K((x_1,i),(x_2,j)) = \sum_{k=1}^{N} d_k K_X^k(x_1,x_2) K_Y^k(i,j),$$

that includes the possibility of optimizing the matrix $\mathbf{L}$ in (1) as a conic combination of basis matrices. In principle, proper complexity control allows to combine an arbitrarily large, even infinite [2], number of kernels. However, computational and memory constraints force the user to specify a relatively small dictionary of basis kernels to be combined, which again calls for a certain amount of domain knowledge.

2 Output Kernel Learning

A more direct approach to synthesize the output kernel from the data consists in solving a two-level regularization problem of the form

$$\min_{\mathbf{L}\in\mathbb{S}_+} \left[\min_{f\in\mathcal{H}_\mathbf{L}} \left(\sum_{j=1}^{m} \sum_{i=1}^{\ell_j} V(y_{ij}, f_j(x_{ij})) + \|f\|^2_{\mathcal{H}_\mathbf{L}} + \Omega(\mathbf{L}) \right) \right],$$

where V is a suitable loss function, $\mathcal{H}_\mathbf{L}$ is the Reproducing Kernel Hilbert Space of vector-valued functions associated with the reproducing kernel (1), Ω is a suitable matrix regularizer, and $\mathbb{S}_+$ is the cone of symmetric and positive semidefinite matrices. We call such an approach Output Kernel Learning (OKL).

A technique of this kind was introduced in [3] for the case where V is a square loss function, Ω is the squared Frobenius norm, and the input data x_{ij} are the same for all the output components f_j. Such special structure of the objective functional allows to develop an effective block coordinate descent strategy where each step involves the solution of a Sylvester linear matrix equation. Regularizing the output kernel with a squared Frobenius norm leads to a simple and effective computational scheme. However, we may want to encourage different types of relationship structures in the output space. Along this line, [4] introduces low-rank OKL, a method to discover relevant low dimensional subspaces of the output space by learning a low-rank kernel matrix. This is achieved by regularizing the output kernel with a combination of the trace and a rank indicator function, namely

$$\Omega(\mathbf{L}) = \mathrm{tr}(\mathbf{L}) + I(\mathrm{rank}(\mathbf{L}) \leq p).$$

For $p = m$, the hard-rank constraint disappears and Ω reduces to the trace norm which, as it is well known, encourages low-rank solutions. Setting $p < m$ gives up convexity of the regularizer but, on the other hand, allows to set a hard bound on the rank of the output kernel, which can be useful for both computational and interpretative reasons. Low-rank OKL enjoys interesting properties and interpretations. Just as sparse MKL with a square loss can be seen as a nonlinear generalization of (grouped) Lasso, low-rank OKL is a natural kernel-based generalization of reduced-rank regression, a popular multivariate technique in statistics.

For problems where the inputs x_{ij} are the same for all the tasks, optimization for low-rank OKL can be performed by means of a rather effective procedure that iteratively computes eigendecompositions. Importantly, the size of the involved matrices can be controlled by selecting the parameter p. Unfortunately, more general multi-task learning problems where each task is sampled in correspondence with different inputs require completely different methods. If a square loss is adopted, it turns out that an effective strategy to approach these problems consists in iteratively apply inexact Preconditioned Conjugate Gradient (PCG) solvers [5] to suitable linear operator equations that arise from the optimality conditions.

3 Concluding remarks and future directions

Learning output kernels via regularization is an effective way to solve multi-task learning problems where the relationships between the tasks are highly uncertain. The OKL framework that we have sketched in the previous section is rather general and can be developed in various directions. Effective optimization techniques for more general (non-quadratic) loss functions are still lacking and the use of a variety of matrix penalties for the output kernel matrix is yet to be explored.

References

[1] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[2] A. Argyriou, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In Peter Auer and Ron Meir, editors, *Learning Theory*, volume 3559 of *Lecture Notes in Computer Science*, pages 338–352. Springer Berlin / Heidelberg, 2005.

[3] F. Dinuzzo, C. S. Ong, P. Gehler, and G. Pillonetto. Learning output kernels with block coordinate descent. In *Proceedings of the 28th Annual International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

[4] F. Dinuzzo and K. Fukumizu. Learning low-rank output kernels. *Journal of Machine Learning Research - Proceedings Track*, 20:181–196, 2011.

[5] F. Dinuzzo. Learning output kernels for multi-task problems. *Neurocomputing*, (to appear), 2013.