Inductive Transfer using Kernel Multitask Latent Analysis

Z. XIANG and K. P. BENNETT
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY, USA
xiangz@rpi.edu, bennek@rpi.edu

Abstract

We develop the Kernel Multitask Latent Analysis (KMLA) method for modeling many-to-many relationships between inputs and responses, and show how it can be applied to inductive transfer problems in bioseparations. KMLA performs dimensionality reduction targeted towards a multitask loss function much like Kernel Partial Least Squares (KPLS). KPLS is limited to least squares multiple regression while KMLA is a more general approach that can utilize widely-used convex loss functions for inference tasks. KMLA achieves inductive transfer between tasks by forcing the tasks to share the same latent features. In the bioseparation problem, the goal is to predict the retention times for a novel anion-chromatography system; only a few retention times are known for the target systems, while many protein retention times are known for the related systems. KMLA is used with semi-supervised loss functions that do not require that all proteins have responses for all the systems. Results are presented for both regression and ranking losses. KMLA significantly outperforms both single task KMLA and KPLS, and the existing missing response algorithm for multitask PLS extended to kernels.

1 Introduction

We focus on how a novel kernel method for constructing many to many mappings can be used for inductive transfer. Prior work has demonstrated that multitask learning (MTL), e.g. training related tasks in parallel, can improve generalization. Learning methods that can perform many-to-many mappings using some sort of task relatedness can be used for multitask learning. In MTL for neural networks, the multiple tasks share common hidden units [2]. MTL kernel learning methods enforce task relatedness by using an appropriate joint capacity control such as such as minimizing the variance of the weights of the SVM for each task [4]. Partial least squares (PLS) [5], a spectral regression method from statistics, has been successfully used for inductive transfer for industrial applications [9]. In PLS, the multiple tasks are modeled using a common set of latent features consisting of linear combinations of the original variables.

Kernel Multitask Latent Analysis (KMLA) is a general kernel method for constructing many-to-many mappings for arbitrary loss functions similar to PLS and single-task Boosted

Latent Features [8]. KMLA uses dimensionality reduction for capacity control and task relatedness. For least squares loss, KMLA reduces to KPLS. KMLA's predictive functions are linear combinations of orthogonal latent features in the kernel defined feature space targeted to a given loss function. Here we use KMLA with a semi-supervised loss function that does not require the responses to be known for all tasks.

We demonstrate how the proposed KMLA approach can be used for MTL problems in bioseparation. The development of efficient bioseparation processes for the production of high-purity biopharmaceuticals is one of the most pressing challenges facing the pharmaceutical and biotechnology industries. Developing elution or displacement systems to remove closely related impurities often requires a significant amount of experimentation to find the proper combination of stationary phase material, salt type and/or displacers to achieve sufficient selectivity and productivity in these separation techniques. Thus the multiple tasks are to model retention times of ion-exchange chromatography systems that vary in displacers and salt types used [11, 10, 7].

2 Multitask Latent Analysis

For simplicity, we use linear MLA instead of KMLA for illustration here. KMLA is the nonlinear extension to MLA using the kernel defined feature space instead of the original input space. MLA constructs a reduced-rank linear function that minimizes or approximately minimizes the loss function:

$$\min_{\mathbf{B},\boldsymbol{\mu}} \quad Loss(\mathbf{Y}, \mathbf{XB} + \mathbf{e}\boldsymbol{\mu}) \tag{1}$$

for a given set of data with input $\mathbf{X} \in R^{m \times n}$ and k responses (one for each of k tasks) $\mathbf{Y} \in R^m \times k$, and loss function $Loss : R^{m \times n} \rightarrow R^{m \times k}$. The row vector $\boldsymbol{\mu}$ accounts for any linear shifts in the response. Like reduced-rank regression, PLS and BLF, MLA regularizes the solution by restricting $\mathbf{B}$ to be rank lat and by forcing extracted latent variables to be orthogonal. It constructs lat number of linear orthogonal latent variables, $\mathbf{T} = \mathbf{XA}$ with $\mathbf{T}'\mathbf{T} = \mathbf{I}$, and makes a prediction coefficient matrix $\mathbf{B}$ such that $\mathbf{B} = \mathbf{AC}'$, which approximately solves Equation (1). Adopting the proof in [8], one can show, MLA solves (1) in a finite number of iterations. Regularization and task relatedness is achieved by forcing the problems to use the shared set of latent features, $\mathbf{T}$.

We assume that the loss function is a *fully separable* convex differentiable or subdifferentiable loss function. This class of loss functions is very large, including convex loss functions commonly used for regression, classification, novelty-detection, and ranking. Fully separable means that the loss function can be written as a sum of $m \times k$ individual errors for each observation and response. $\mathbf{Y}$ is the original responses and $\mathbf{F}$ is the predicted responses. $Loss$ denotes the MTL loss function, and L denotes its components. The loss for each point i and task j has a given weight $\delta_{ij} \geq 0$ allowing points and tasks to have different weights. Setting $\delta_{ij} = 0$ effectively means the response of point i on task j is unknown, allowing the training data to be used for different tasks. Formally the loss function is $Loss(\mathbf{Y}, \mathbf{F}) = \sum_{i=1}^{m} \sum_{j=1}^{k} \delta_{ij} L(\mathbf{Y}_{ij}, \mathbf{F}_{ij})$. The function L can be any appropriate error measure for a data point or pairs of data points. In a slight abuse of notation we will let ∇ and ∂ reference gradients or appropriate subgradients, e.g. $(\nabla Loss(\mathbf{Y}, \mathbf{F}))_{ij} = \frac{\delta_{ij} \partial L(\mathbf{Y}_{ij}, \mathbf{F}_{ij})}{\partial \mathbf{F}_{ij}}$.

MLA is subspace algorithm closely related to a nonlinear conjugate gradient algorithm. Alternatively, one can think of it as a multitask generalization of the BLF [8] boosting algorithm that achieves inductive transfer by forcing the tasks to be linear combinations of on the same set of weak hypotheses. In MLA, the weak hypothesis are linear functions in feature space that are explicitly forced to be orthogonal. Thus MLA is a dimensionality reduction algorithm. One can prove that MLA converges in at most rank k iterations to a

solution of the original problem and demonstrate empirically the convergence rate is much faster MLA then a gradient descent boosting algorithm. The orthogonality is achieved by deflation. Linear MLA is given in the appendix. Like BLF, MLA is readily to be kernelized and allows the latent features to be nonlinear.

3 Protein Retention Times

We predict the protein retention times for ion-exchange chromatography systems. The goal is to predict the retention times of one system given the results for 17 different proteins on 5 other systems and 10 proteins on this system. There are 10 descriptors for each protein. The input $\mathbf{X}$ is a 17×10 matrix and the output $\mathbf{Y}$ is a 17×6 matrix. The descriptors are calculated using RECON (an algorithm for the rapid reconstruction of molecular charge densities and charge density-based electronic properties of molecules) [1] and drug discovery software Molecular Operating Environment (MOE) [3]. Each protein has six retention times measured in systems under the combinations of two resins (SOURCE and FFSEPH) and three buffer salts (Na+, Ca2+, NH4+). Thus the six responses/tasks (measured in minutes) are: FFSEPH-NA (R1), SOURCE-NA (R2), FFSEPH-CA (R3), SOURCE-CA (R4), FFSEPH-NH4 (R5) and SOURCE-NH4 (R6). See [10, 7] for more details.

The experiments were designed to demonstrate the inductive transfer of five tasks to a sixth target task. For the target response, the training response consists of 10 randomly selected points for the target data and the remaining 7 proteins are used for test. Then the task is to accurately predict the retention times of the test target set using all the proteins, the 10 known retention times for the target task, and all the retention times for the nontarget tasks. The experiment was repeated for each task for 20 different training and testing partitions and the average errors or accuracies are reported. Since the number of training points is extremely small, we set the number of latent features to 2 for all multi-task algorithms and 1 for single-task algorithms.

The first experiment used KMLA with least squares regression loss with $\delta_{ij} = 0$ for test proteins i in the target task j and $\delta_{ij} = 1$ otherwise. Figure 1 provides sample results for the Gaussian kernel. KMLA was compared to single task KPLS trained with just the 10 labelled points. In addition we compare it with multitask KPLS with the test responses treated as missing values using the approach of [9] generalized to kernels. During training, multitask KPLS estimates each missing test responses for point i and task j as the mean of the average known response for point i and the average known response for task j. The Q-squared statistic on the testing data is reported. $Q^2 = \frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2}$ (y_i is the real value and $\hat{y}_i$ is the prediction) and Q^2 near 0 is better. One can clearly see that using multitask learning with either KPLS missing or KMLA is a dramatic improvement over single task learning, with KMLA being the preferred approach.

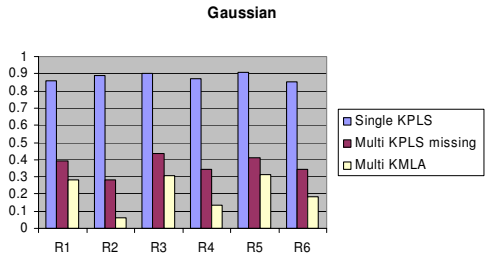

Figure 1: Results for KPLS and KMLA with 10 training points on target task and 17 training points on on other tasks

Table 1: Single mode and Multi-mode KMLA ranking loss: Kendall's τ vs. different training size (Gaussian kernel $\sigma = 0.5$ and Fac=2)

	Multi	task			Single	task		
TrSz	5	8	11	14	5	8	11	14
Res1	0.6924	0.6944	0.7267	0.9333	0.2652	0.3778	0.44	0.5
Res2	0.8424	0.8333	0.8467	0.9333	0.3167	0.4806	0.5267	0.5333
Res3	0.6136	0.7444	0.7333	0.8333	0.2621	0.4722	0.5	0.5
Res4	0.7227	0.8	0.78	0.8667	0.303	0.5	0.5133	0.6667
Res5	0.753	0.7722	0.8267	0.8333	0.2621	0.4361	0.46	0.5333
Res6	0.8182	0.8139	0.86	0.9333	0.3076	0.4667	0.52	0.5333

We also treated the chromatography problem as a ranking problem. We examined each pair of proteins and the protein with higher retention time was considered to be higher ranked. KMLA was extended to use the paired ranking loss used in the support vector machine type model in [6]. For ranking, Kendall's τ for the testing data is computed to compare methods, with results near 1 as best. The same parameters were used as for the regression case. Table 1 shows results for different training set sizes of the target response. The remaining proteins for the target response are considered to be the testing data. Since KPLS is limited to regression, results are shown for KMLA trained on a single tasks and KMLA trained on multiple tasks. The results show that multitask learning outperforms single task learning and that the faster rise in the learning curve for multitask KMLA over single task KMLA indicates inductive transfer.

4 Discussion

KMLA provides a flexible nonlinear semi-supervised learning framework that builds on the PLS algorithm already successfully used for MTL. We provide here sample results for a chromatography problem with six responses treated both as a regression and ranking problem. The results show dramatic increases in accuracy for MTL versus single task. Much less data is required to achieve the same or better accuracy. Space does not permit us to present the more extensive testing on this and another chromatography problem which further support these conclusions. The KMLA algorithm has the advantage that it can be used in both semi-supervised and supervised models and it is computationally efficient. Like principal components or canonical correlation analysis, KMLA yields latent features that can be used for visualization or as features in other learning algorithms, but KMLA's features are targeted to a specific loss function and learning task. Note KMLA does not require the responses for each task to be known for all training points. Exploiting this property, we also have developed an efficient method for selecting the number of latent features and kernel parameters based on leaving out data in the non-target tasks with more data. KMLA is an efficient greedy algorithm. An open question is whether seeking globally optimal solutions to the reduced rank loss problem would yield better results.

References

[1] C.M. Breneman and T.R. Thompson. Electron density modeling of large systems using the transferable atom equivalent method. *Computers and Chemistry*, 19(3):161, 1995.

[2] R. Caruana. Multi-task learning. *Machine Learning*, 28:41–75, 1997.

[3] CCG. drug discovery software—molecular operating environment (moe), 2000. `http://www.chemcomp.com/`.

[4] T. Evegniou and M.Pontil. Regularized multi–task learning. In *Proceedings of 17-th SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD)*, 2004.

[5] A. Höskuldsson. PLS regression methods. *Journal of Chemometrics*, 2:211–228, 1988.

[6] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[7] A. Ladiwala, K. Rege, C.M. Breneman, and S.M. Cramer. Investigation of mobile phase salt type effects on protein retention and selectivity in cation-exchange systems using quantitative structure retention relationship models. *Langmiur*, 19(20):8443–8454, 2003.

[8] M. Momma and K.P. Bennett. Constructing orthogoal latent features for arbitrary loss. In M. Nikravesh I. Guyon, S. Gunn and L. Zadeh, editors, *Feature Extraction, Foundations and Applications*, 2006. Springer, to appear.

[9] S. Rnnar, P. Geladi, F. Lindgren, and S. Wold. A pls kernel algorithm for data sets with many variables and few objects. part ii: Cross-validation, missing data and examples. *Journal of Chemometrics*, 9:459–470, 1995.

[10] M. Song, C.M. Breneman, J. Bi, N. Sukumar, and K.P. Bennett etc. Prediction of protein retention times in anion-exchange chromatography systems using support vector regression. *Journal of Chemical Information and Computer Sciences*, 42(6):1347–1357, 2002.

[11] N. Tugcu, A. Ladiwala, C.M. Breneman, and S. M. Cramer. Identification of chemically selective displacers using parallel batch screening experiments and quantitative structure efficacy relationship models. *Analytical Chemistry*, 75,(21):5806–5816, 2003.

Appendix

Algorithm 1 The Multitask Latent Analysis Algorithm

Given $\mathbf{X}$, $\mathbf{Y}$, $Loss$, and lat (the number of LV)

1. Let $\boldsymbol{\mu} \in argmin_{\boldsymbol{\mu}} Loss(\mathbf{Y}, \mathbf{e}\boldsymbol{\mu})$ and $\mathbf{F}^0 = \mathbf{e}\boldsymbol{\mu}$ where $\mathbf{e}$ is a R^m vector of ones and $\boldsymbol{\mu}$ is a row vector
2. Get the first negative gradient: $\mathbf{D}^1 = -\nabla Loss(\mathbf{Y}, \mathbf{F}^0)$
3. For i=1 : lat
4. Compute best rank-one descent direction using SVD or power method:

$$\mathbf{w}_i, \mathbf{s} \in argmin_{\mathbf{w},\mathbf{s}} \frac{\mathbf{D}^i \cdot \mathbf{X}\mathbf{w}\mathbf{s}'}{\|\mathbf{w}\|\|\mathbf{s}\|} \tag{2}$$

5. Save projection $\mathbf{W} = [\mathbf{W}\ \mathbf{w}_i]$.
6. Compute latent variable $\mathbf{t}_i = \mathbf{X}^i\mathbf{w}_i$, $\mathbf{t}_i = \frac{\mathbf{t}_i}{\|\mathbf{t}_i\|}$, $\mathbf{T} = [\mathbf{T}\ \mathbf{t}_i]$
7. Deflate: $\mathbf{p}_i = \mathbf{X}_i{}'\mathbf{t}_i$,$\mathbf{P} = [\mathbf{P}\ \mathbf{p}_i]$, $\mathbf{X}^{i+1} = \mathbf{X}^i - \mathbf{t}_i\mathbf{p}_i{}'$
8. Refit $\mathbf{C}, \hat{\boldsymbol{\mu}}$: $(\mathbf{C}, \hat{\boldsymbol{\mu}}) \in argmin_{\mathbf{C},\hat{\boldsymbol{\mu}}} Loss(\mathbf{Y}, \mathbf{T}\mathbf{C}' + \mathbf{e}\hat{\boldsymbol{\mu}})$
9.
10. Compute negative gradient matrix: $\mathbf{D}^{i+1} = -\nabla Loss(\mathbf{Y}, \mathbf{F}^i)$
11. If $\mathbf{X}'\mathbf{D}^{i+1} = 0$, break because optimal
12. End
13. Compute weight coefficients $\mathbf{A}$ in the original space: $\mathbf{A} = \mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}$, and construct final features: $\mathbf{T}(\mathbf{X}) = \mathbf{X}\mathbf{A} = \mathbf{X}\mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}$
14. Compute prediction coefficients $\mathbf{B}$ in the original (undeflated) space: $\mathbf{B} = \mathbf{A}\mathbf{C}' = \mathbf{W}(\mathbf{P}'\mathbf{W})^{-1}\mathbf{C}'$
15. Final