Multi-Task Learning Using Shared and Task Specific Information

P.K. Srijith and Shirish Shevade*

Computer Science and Automation
Indian Institute of Science, Bangalore
{srijith,shirish}@csa.iisc.ernet.in

Abstract. Multi-task learning solves multiple related learning problems simultaneously by sharing some common structure for improved generalization performance of each task. We propose a novel approach to multi-task learning which captures task similarity through a shared basis vector set. The variability across tasks is captured through task specific basis vector set. We use sparse support vector machine (SVM) algorithm to select the basis vector sets for the tasks. The approach results in a sparse model where the prediction is done using very few examples. The effectiveness of our approach is demonstrated through experiments on synthetic and real multi-task datasets.

Keywords: Multi-task learning, Support vector machines, Kernel methods, Sparse models.

1 Introduction

Multi-task learning (MTL) is used in situations where one has to solve several related learning problems. Multi-task learning models each learning problem as a separate task but instead of learning the tasks independently, learns them together [1]. It is extremely effective when each learning problem is associated with a limited dataset. It enables a task to be learnt using the data from multiple related tasks. This results in a better predictive performance of the individual tasks. It has been shown that multi-task learning performs better than learning tasks independently [2,3,4]. Multi-task learning methods are successfully applied to applications like user preference modeling [5] and conjoint analysis [6].

Multi-task learning (MTL) has recently created a lot of interest in the machine learning community. Many approaches have been proposed to effectively learn multiple related tasks. Task similarity could be captured by restricting different task functions to be close to each other in some distance measure [4]. Bayesian approaches [5] capture task similarity by sharing a common prior among different tasks. Other approaches capture task similarity by sharing a common internal representation [2,3] across all the tasks. Most of the kernel based multi-task learning approaches [4] use all the training examples to make a prediction, resulting in higher computational and storage requirements. Also, they try to capture only task similarity and not task variability.

* The author gratefully acknowledges the support for this work from Infosys Ltd., India.

T. Huang et al. (Eds.): ICONIP 2012, Part III, LNCS 7665, pp. 125–132, 2012.

We propose a novel multi-task learning model in which task similarity is captured by sharing a common set of basis vectors across all tasks. In addition, it has task specific basis vectors which capture the variability across different tasks. It uses both the common set and the task specific set to make predictions for the test data. The use of both the common set and the task specific set helps it to capture the relatedness between tasks more effectively. The basis vector sets are learnt by extending the sparse SVM algorithm [7] for single task learning to the multi-task scenario. It results in a sparse model which requires very few training examples to make a prediction. This enables it to make predictions much faster and is extremely useful when dataset size is large. Experimental results on synthetic and real datasets show the usefulness of our approach.

We discuss some related work in section 2. Section 3 discusses the proposed sparse multi-task learning approach in detail. Section 4 presents the experimental results of running the proposed approach on synthetic and real multi-task datasets. Finally we conclude in section 5.

2 Related Work

We consider multi-task learning problems with T tasks. Each task t is associated with a dataset D_t with m_t examples, *i.e.* $D_t = \{x_{ti}, y_{ti}\}_{i=1}^{m_t}$. Let $n = \sum_{t=1}^{T} m_t$ be the total number of examples from all the tasks. Each task specific dataset D_t comes from the same input and output space $X \times Y$ where $X \subset \mathcal{R}^d$ and $Y = \{+1, -1\}$ for classification or $Y \subset \mathcal{R}$ for regression. We assume that task specific datasets are associated with a different but related sampling distributions P_t. The goal is to learn T functions $f_1, f_2, \ldots, f_T$ for T tasks such that each task specific function f_t gives good generalization performance.

Regularized multi-task learning [8] captures similarity among tasks by assuming all the task specific functions to be close to each other. The parameters of task specific functions are learnt using the modified SVM framework. The approach uses almost the entire training examples to make a prediction. The proposed sparse multi-task learning approach differs from it in using very few training examples to make a prediction. Radial basis function network for multi-task learning [9] captures similarity by sharing the basis functions across all the tasks. The proposed approach differs from it in having a task specific basis function set in addition to the shared set. It enables the proposed approach to more effectively capture the task relations.

3 Sparse Multi-Task Learning

Sparse multi-task learning approach captures the similarity between tasks by restricting the task specific functions to share some common structure. It models this by assuming the predictive function for a task to have a common part shared by all the tasks and a task specific part particular to the task. It represents the predictive function for a task t as $f_t(x) = w_c.\phi(x) + w_t.\phi(x)$, where ϕ is a feature map which maps the examples from input space X to some reproducing kernel

Hilbert space H with an associated kernel function K. The common part w_c captures the similarity among the tasks, while the task specific part w_t captures the variability across different tasks. We assume that the common part w_c takes the basis function expansion form $w_c = \sum_{c=1}^{N} \alpha_c \phi(x_c)$ where $\phi(x_c)$ is the basis associated with the example x_c in the common basis vector set C of size N and α is the parameter associated with the common set C. Examples in the common basis vector set C could belong to any task. Similarly the task specific part w_t for a task t is represented as $w_t = \sum_{j=1}^{M_t} \beta_{tj} \phi(x_{tj})$, where $\phi(x_{tj})$ is the basis associated with the example x_{tj} in the task specific basis vector set J_t of size M_t and β_t is the parameter associated with the task specific set J_t. Examples in the task specific basis vector set J_t belongs only to task t. Using the basis function expansion form and kernels to represent the inner product between basis functions, the predictive function f_t for a task t could be written as

$$f_t(x) = K_{xC}.\alpha^\top + K_{xJ_t}.\beta_t^\top \tag{1}$$

where $K_{xC} = [K(x, x_1), \ldots, K(x, x_N)]$, $\alpha = [\alpha_1, \ldots, \alpha_N]$, $K_{xJ_t} = [K(x, x_{t1}), \ldots, K(x, x_{tM_t})]$, $\beta_t = [\beta_{t1}, \ldots, \beta_{tM_t}]$ and $K(x_i, x_j) = \phi(x_i).\phi(x_j)$.

The selection of basis vector sets and the estimation of parameters for multi-task classification problems are done by minimizing the objective function

$$\underset{\alpha,\beta_1,\ldots,\beta_T,C,J_1,\ldots,J_T}{\arg\min} \frac{\gamma}{2} \alpha K_{CC} \alpha^\top + \frac{\lambda}{2} \sum_{t=1}^{T} \beta_t K_{J_t J_t} \beta_t^\top + \frac{1}{2} \sum_{t=1}^{T} \sum_{i \in I_t} (1 - y_{ti} o_{ti})^2 \tag{2}$$

where γ and λ are regularization parameters, K_{CC} is the kernel matrix formed from examples in the common set C, $K_{J_t J_t}$ is the kernel matrix formed from examples in the task set J_t, $o_{ti} = K_{iC}\alpha^\top + K_{iJ_t}\beta_t^\top$ is the output of the i^{th} example belonging to task t, and $I_t = \{i : 1 - y_{ti} o_{ti} > 0\}$.

The regularization parameter controls the common and task specific basis vector sizes. A low value of γ relative to λ selects more common basis vectors than task specific basis vectors. This is ideal for the situations in which tasks are similar to each other. In the limit when $\frac{\gamma}{\lambda}$ tends to 0, this is equivalent to combined task learning in which a single classifier is learned by pooling together data from all the tasks. But in situations where tasks are dissimilar it is appropriate to choose a low value for λ relative to γ resulting in the selection of more task specific basis vectors than common basis vectors. In the limit when $\frac{\lambda}{\gamma}$ tends to 0,this is equivalent to single task learning in which tasks are learnt independently using their respective datasets.

The basis vector sets and the parameters are obtained by extending the sparse SVM [7] approach for single task learning to the multi-task learning scenario. We select the common basis vector set and task basis vector sets incrementally. After each basis selection we re-estimate the common and task parameters. Section 3.1 discusses the procedure to estimate the parameters assuming we have selected common basis vector set C and task basis vector sets J_t's. Section 3.2 discusses the procedure to select the common and task basis vector sets.

3.1 Parameter Estimation

Sparse MTL approach needs to estimate $T+1$ parameters, one common parameter α and T task parameters β_t. We use an alternative optimization approach to estimate the parameters. The approach minimizes the objective function (2) with respect to one of the parameter keeping others fixed. This is repeated for each parameter and the entire procedure is continued until the relative decrease in objective function value becomes small. Algorithm (1) describes the parameter estimation procedure in detail.

Algorithm 1
Procedure *Parameter Estimation*
Input: $C, J_1, \ldots, J_T$
Output: α, $\beta_1, \ldots, \beta_T$

1. Set k = 0. Choose suitable starting vectors $\alpha^{(0)}$, and ${\beta_t}^{(0)}$ for each task t.
   repeat
2. For the current values of task parameters(${\beta_t}^{(k)}$), obtain $\alpha^{(k+1)}$ by minimizing the objective function (2) with respect to α using Newton method with line search.
3. **for** each task t
4. For the current value of common parameter($\alpha^{(k+1)}$), obtain ${\beta_t}^{(k+1)}$ by minimizing the objective function (2) with respect to β_t using Newton method with line search.
5. k ←k+1
6. **until** relative decrease in objective function value (2) is small.

The parameter estimation uses the Newton method and it requires the calculation of the gradients and the generalized Hessians of the objective function (2). The gradients and the generalized Hessians of the objective function (2) with respect to the parameters α and β_t's are

$$g_\alpha = \gamma K_{CC}\alpha^\top - \textstyle\sum_{t=1}^{T} K_{CI_t}[y_{I_t} - o_{I_t}] \qquad P_\alpha = \gamma K_{CC} + \displaystyle\sum_{t=1}^{T} K_{CI_t}K_{I_tC}$$
$$g_{\beta_t} = \lambda K_{J_tJ_t}\beta_t^\top - K_{J_tI_t}[y_{I_t} - o_{I_t}] \qquad P_{\beta_t} = \lambda K_{J_tJ_t} + K_{J_tI_t}K_{I_tJ_t} \forall t \quad (3)$$

Here g and P denote the gradient and the generalized Hessian respectively with respect to the parameters denoted by the subscripts, y_{I_t} is the column vector of labels from task t indexed by I_t, and o_{I_t} is the column vector of outputs from task t indexed by I_t.

3.2 Basis Vector Selection

The basis vectors are selected greedily in an incremental mode. The selection involves adding basis vectors to the common set and to T task specific sets. The basis vectors for the common set are obtained from examples from all the tasks while basis vectors for the task specific set are obtained from the task specific

examples. Common basis vectors are selected first and task specific basis vectors are selected later. The basis vectors are selected until the relative decrease in the objective function value becomes small. Alternatively, one can predefine the number of basis vectors to be selected. The basis vector selection procedure results in a sparse model with very few elements in the common set and task specific sets. Algorithm 2 describes the basis vector selection procedure in detail.

Algorithm 2
Procedure *Basis Vector Selection*
1. **repeat**
2. Select a basis vector from the complete training data.
3. Add the selected basis vector to the common set C.
4. Perform parameter estimation using Algorithm 1.
5. **until** relative decrease in objective function value is small
6. **for** each task t
7. **repeat**
8. Select a basis vector from the training dataset D_t.
9. Add the selected basis vector to the task specific set J_t.
10. Perform parameter estimation using Algorithm 1.
11. **until** relative decrease in objective function value is small

During basis vector addition, a basis vector is selected from the training set which results in maximum decrease in objective function value on addition of it to the basis set. Selecting a basis vector from the entire training set is time consuming. Hence the basis vector is selected from a candidate set of size κ containing κ examples selected randomly from the training set. During basis selection, the objective function is optimized only with respect to the parameter corresponding to the newly added basis vector. In this case the objective function is a simple quadratic function in the variable of optimization. Therefore the optimization variable and the decrease in the objective function value can be calculated analytically. Let the newly added basis vector to the common set be c and the parameter corresponding to it be α_c. Then α_c and the reduction in the objective value is given by $-g_{\alpha_c}/P_{\alpha_c}$ and $g^2_{\alpha_c}/P_{\alpha_c}$ respectively, where

$$g_{\alpha_c} = \gamma K_{cC}\alpha - \sum_{t=1}^{T}\sum_{i\in I_t} K_{cI_t}(y_{I_t} - o_{I_t}) \qquad P_{\alpha_c} = \gamma K_{cc} + \sum_{t=1}^{T} K_{cI_t}K_{I_tc} \quad (4)$$

We could obtain similar expressions for task specific basis vector selection also. After each basis vector addition we obtain new values of the parameters by following the parameter estimation procedure described in section 3.1.

Newton method and basis vector selection require modifications in the generalized Hessian due to changes in the sets I_t, C and J_t. It could be done cheaply by maintaining a Cholesky decomposition of the generalized Hessian [7] and using efficient rank one updates [10]. Let the number of training examples in each task specific dataset be m and total number of examples be n ($n = Tm$). Let current number of elements in the common basis vector set be N and task specific basis

vector set be M. On addition of a new common basis vector the cost incurred for computing new elements of the generalized Hessian and maintaining its Cholesky decomposition are $\mathcal{O}(TmN)$ and $\mathcal{O}(N^2)$ respectively. Assuming $N \ll n$ the cost of a single common basis vector addition is $\mathcal{O}(\kappa nN)$. Setting maximum number of common basis vectors to N_{max}, common basis vector set selection takes $\mathcal{O}(\kappa nN_{max}^2)$ time. Similarly the addition of a single task specific basis vector takes $\mathcal{O}(\kappa mM)$ time(assuming $M < m$). Setting maximum number of task specific basis vectors to M_{max}, task specific basis vector selection takes $\mathcal{O}(\kappa mM_{max}^2)$ time. For all tasks it takes $\mathcal{O}(\kappa TmM_{max}^2) = \mathcal{O}(\kappa nM_{max}^2)$ time.Hence the time complexity of the proposed approach is $\mathcal{O}(\kappa nN_{max}^2) + \mathcal{O}(\kappa nM_{max}^2)$.

Multi-task regression problems are solved in a similar way to the classification problems. The difference comes in the objective function which uses a least squares loss function instead of the squared hinge loss function used in the multi-task classification problem (5). Multi-task regression problem minimizes the following objective function.

$$\underset{\alpha,\beta_1,\ldots,\beta_t,C,J_1,\ldots,J_T}{\arg\min} \frac{\gamma}{2}\alpha K_{CC}\alpha^\top + \frac{\lambda}{2}\sum_{t=1}^{T}\beta_t K_{J_tJ_t}\beta_t^\top + \frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{m_t}(y_{ti} - o_{ti})^2 \quad (5)$$

The calculation of gradients and generalized Hessians for the objective function (5) is similar to (3) except that I_t contains the entire training examples from task t.

4 Experiments

We conduct experiments for both multi-task classification and regression problems. Classification experiments are conducted on a synthetic dataset. Regression experiments are done on a real dataset. We compare the proposed sparse multi-task learning model (sparseMTL) against regularized multi-task learning (regMTL) [4], combined task learning (sparseCTL) and single task learning (sparseSTL). SparseCTL is learnt by pooling together data from all the tasks and then learning a single model on the combined dataset using sparse SVM [7]. The results reported for sparseCTL use the same number of basis vectors as sparseMTL. In sparseSTL each task is learnt independently using sparse SVM on the dataset corresponding to that task. The results reported for sparseSTL use the entire training dataset corresponding to the task as the basis vector set. All the experiments use the Gaussian kernel, $K(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma_d^2}\right)$.

4.1 Multi-task Classification

Multi-task classification experiments are done on a synthetic dataset. The synthetic data models the preferences of individuals while choosing products. The dataset is simulated as described in [4]. The synthetic data consists of 30 tasks. Every task is associated with 96 training examples and 96 test examples. In total

Table 1. Mean misclassification error and mean number of basis vectors (given in brackets) for regMTL, sparseMTL, sparseSTL and sparseCTL on the synthetic data. Bolded column indicates the best result. We also report mean number of common basis vectors and task basis vectors obtained for sparseMTL on the synthetic data.

Similarity	RegMTL	SparseMTL	SparseCTL	SparseSTL	Similarity	Common	Task
High	8.16%(964)	**7.22%**(60)	15.59%(60)	10.49%(96)	High	18	42
Low	9.79%(1330)	**9.40%**(70)	29.06%(70)	10.44%(96)	Low	12	58

there are 2880 training and 2880 test examples. We consider two kinds of synthetic data, one in which tasks are less similar and the other in which tasks are more similar. For both the cases we consider synthetic datasets with low noise. Table 1 shows the mean misclassification error of different approaches over 5 independent runs on the dataset. It also reports the mean number of common and task basis vectors obtained for sparseMTL. Each approach is run for different hyper-parameter value settings and the best among those is reported.

We could observe from Table 1 that sparseMTL gives the best result for both the high similarity and the low similarity dataset. In addition the sparseMTL approach provides an advantage in terms of number of basis vectors needed for prediction. SparseMTL requires an order of magnitude less number of basis vectors than regMTL and performs better than regMTL. In the low similarity case sparseMTL is found to select relatively less number of common basis vectors and more number of task specific basis vectors in order to capture the dissimilarity among tasks . In the high similarity case, it is found to select relatively more number of common basis vectors and less number of task specific basis vectors.

4.2 Multi-task Regression

Multi-task regression experiments are done on school dataset[4]. The dataset consists of examination records of 15362 students over 139 schools. Each student record has 27 dimensions and the number of student records associated with each school varies from 20-150. The goal is to predict exam scores of students from each school. We used 75% of the examples from each task as the training set and the remaining 25% as the test set. In total the training data contains 11472 examples and the test data contains 3890 examples. The performance metric used is the explained variance[4] which is defined as $1 - \frac{\text{sum squared error}}{\text{total variance}}$. Table 2 reports the mean explained variance and the mean number of basis vectors required for regMTL, sparseMTL, sparseSTL, and sparseCTL over 10 independent runs on the school dataset. It also reports the mean number of common and task basis vectors selected by sparseMTL.

We could observe from Table 2 that sparseMTL performance is marginally better than regMTL. More importantly it could achieve this performance with very less number of basis vectors. SparseMTL is found to select more number of common basis vectors than task basis vectors capturing the high similarity among the tasks in the school dataset.

Table 2. Mean explained variance and mean number of basis vectors (given in brackets) for regMTL, sparseMTL, sparseSTL and sparseCTL on the school data. Bolded column indicates the best result among the different approaches. In sparseSTL every task uses the entire training data corresponding to it as the basis vector set and its size varies across tasks. We also report the mean number of common and task basis vectors selected by sparseMTL on the school dataset.

RegMTL	SparseMTL	SparseCTL	SparseSTL
0.3275(11330)	**0.3282**(75)	0.2833(75)	0.2710

Total	Common	Task
75	65	10

5 Conclusion

We proposed a novel approach to multi-task learning which captures the task relationships through common and task specific basis vector sets. We also developed an approach to select the basis vector sets. It resulted in a sparse multi-task model which uses very few training examples for predictions. The sparse model can handle very large datasets and makes predictions faster. Experimental results showed that the proposed approach was able to achieve the generalization performance close to that achieved by other multi-task approaches with very few number of basis vectors. The proposed approach, however, is not directly applicable to multi-label classification problems since the tasks share the dataset.

References

1. Caruana, R.: Multitask Learning. Machine Learning 28(1), 41–75 (1997)
2. Ando, R.K., Zhang, T.: A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. J. Mach. Learn. Res. 6, 1817–1853 (2005)
3. Bakker, B., Heskes, T.: Task Clustering and Gating for Bayesian Multitask Learning. J. Mach. Learn. Res. 4, 83–99 (2003)
4. Evgeniou, T., Micchelli, C.A., Pontil, M.: Learning Multiple Tasks with Kernel Methods. J. Mach. Learn. Res. 6, 615–637 (2005)
5. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task Learning for Classification with Dirichlet Process Priors. J. Mach. Learn. Res. 8, 35–63 (2007)
6. Argyriou, A., Evgeniou, T., Pontil, M.: Convex Multi-task Feature Learning. Machine Learning 73(3), 243–272 (2008)
7. Keerthi, S.S., Chapelle, O., DeCoste, D.: Building Support Vector Machines with Reduced Classifier Complexity. J. Mach. Learn. Res. 7, 1493–1515 (2006)
8. Evgeniou, T., Pontil, M.: Regularized Multi-task Learning. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 109–117. ACM (2004)
9. Liao, X., Carin, L.: Radial Basis Function Network for Multi-task Learning. In: Advances in Neural Information Processing Systems 18, pp. 795–802. MIT Press (2006)
10. Seeger, M.: Low Rank Updates for the Cholesky Decomposition. Technical report, University of California at Berkeley (2004)