Multitask Learning using Transformation Functions

Arvind Agarwal **Hal Daumé III**
School of Computing
University of Utah
{arvind,hal}@cs.utah.edu

1 Introduction

Recently, there has been a significant interest in Multi-Task Learning (MTL) where multiple tasks are learned simultaneously to achieve better learning models. It has been shown [3, 4, 7] that learning multiple tasks together help the learning given that, these tasks are related and one is able to use the right notion of task relatedness.[1] There exist many notions of task relatedness, one of the widely believed is: Two tasks are related if their parameters are close [5, 6]. Our work is motivated by a simple counter example of this notion.

Consider a binary classification task t that classifies the given data into two categories. Let w be the optimal weight vector of the linear classifier associated with this task. Now construct an artificial binary classification task u by flipping the labels of the given data; the weight vector of the classifier associated with this task u would be $-w$. Now it is obvious that while tasks t and u are highly related (they are basically the same tasks), their parameters are far from each other ($2\|w\|$ distance apart). In this example task t is just a reflection of task u. Similar counter examples can be constructed by choosing different transformations between t and u e.g. translations, rotations. In this work we give an assumption free notion of task relatedness that is based on such transformations, and is naturally derived from the data. Informally,

Two tasks are said to be related if one task can be transformed into other task by applying some transformation function on the learning function associated with the first task or vise versa. The amount of relatedness is measured in terms of the complexity of the transformation function.

2 Approach

We now briefly describe the proposed MTL framework based on the transformation between tasks [1, 2]. We begin with some notations. Let $\mathcal{X}$ be the domain for all tasks[2], $\mathcal{X}_t \subset \mathcal{X}$ be the set of training examples for task t and $\mathcal{Y}_t$ be the set of corresponding class labels. Given such example-label pairs $(\mathcal{X}_t, \mathcal{Y}_t)$, the goal is to learn the function f_t such that for each $x \in \mathcal{X}$, f_t gives the corresponding label. Let T be the total number of tasks and $f_1, \ldots f_T$ be the classification functions for these tasks. Now we define a new set of functions called *transformation functions* between tasks. Let G_{ij} be such a transformation function that transforms the task function f_j to f_i, in particular for each $x \in \mathcal{X}$,

$$f_i(x) = G_{ij}(f_j(x), x).$$

Now under the proposed MTL framework, a classification function f_t should perform well on its own task, and at the same time, be close to the other tasks, when they are transformed i.e. functions f_t and

[1] "task relatedness" has been a major concern in MTL, and there has not been much research in this area. Some of the theoretical work can be found in [1]

[2] This is a fundamental assumption in MTL that data for all tasks come from the same domain, otherwise it becomes a different problem, a mixture of MTL and domain adaptation problems.

Presenter: Arvind Agarwal, Category: Learning Algorithms, Presentation Preference: Oral

$G_{tu}(f_u(x), x)$ should be similar. This intuition suggests the following cost function:

$$\mathcal{C} = \sum_{t=1}^{T} \Big(\sum_{\substack{x \in \mathcal{X}_t \\ y \in \mathcal{Y}_t}} \underbrace{\mathcal{L}(f_t(x), y)}_{\substack{f_t \text{ performs well} \\ \text{on training data of } t}} + \underbrace{\gamma \, ||f_t||^2_{\mathcal{H}}}_{\substack{f_t \text{ generalizes} \\ \text{well on test data}}} + \sum_{u \neq t} \lambda_u \sum_{x \in \mathcal{X}} \underbrace{\mathcal{U}(f_t(x), G_{tu}(f_u(x), x))}_{\substack{f_t \text{ is close to} \\ \text{the transformation of } f_u}} + \underbrace{\eta \, ||G_{tu}||^2_{\mathcal{H}}}_{\substack{G_{tu} \text{ is simple \&} \\ \text{does not overfit}}} \Big) \quad (1)$$

where γ controls the complexity of the task function, and λ trades off the amount of transfer between the tasks t and u. It is to be noted that term belonging to the loss function $\mathcal{L}$ is defined only for the examples that belong to the task t, while the term belonging to the loss function $\mathcal{U}$ is defined over the entire domain $\mathcal{X}$ with no reference to class labels. This allows one to use the unlabeled data in this framework therefore makes it useful for semi-supervised learning. Also note that we have additionally regularized the cost function by $||G_{tu}||^2_{\mathcal{H}}$ in order to prefer the simple transformation functions. We have used the same reproducing kernel Hilbert space (RKHS) for classification functions and transformation functions assuming that RKHS is rich enough to encompass all these functions. Given the cost function (1), proposed MTL framework minimizes (1) for all task functions $f_1, \ldots f_T$, and for all transformation functions G_{tu} i.e. $\min_{f_1 \ldots f_T, G_{tu} \, \forall t,u} \mathcal{C}$.

(1) gives us a learning paradigm where all task functions $f_1, \ldots f_T$ and transformation functions G_{tu} are learned simultaneously. Optimizing (1) simultaneously for all tasks functions and transformation functions is hard, therefore we resort to alternate optimization i.e. learning one function at a time keeping all other functions fixed. This gives us separate cost functions for learning task functions and transformation functions. Cost function for task t is given as:

$$\mathcal{C}_{f_t} = \sum_{\substack{x \in \mathcal{X}_t \\ y \in \mathcal{Y}_t}} \mathcal{L}(f_t(x), y) + \gamma \, ||f_t||^2_{\mathcal{H}} + \sum_{u \neq t} \lambda_u \sum_{x \in \mathcal{X}} \mathcal{U}(f_t(x), G_{tu}(f_u(x), x)) \quad (2)$$

while cost function for learning the transformation functions can be written as:

$$\mathcal{C}_{G_{tu}} = \sum_{x \in \mathcal{X}} \mathcal{U}(f_t(x), G_{tu}(f_u(x), x)) + \eta \, ||G_{tu}||^2_{\mathcal{H}} \, . \quad (3)$$

Solution to both (2) and (3) is given by the representer theorem and is of the form $\sum_{i=1}^{|\mathcal{X}|} \alpha_i k(\cdot, x_i)$ where $k(\cdot, \cdot)$ is the kernel associated with RKHS. Initial experiments on artificial tasks show that we were able to learn the transformation functions therefore exploit the task relatedness to learn the better models.

References

[1] Ben-David, S., and Borbely, R. S. A notion of task relatedness yielding provable multiple-task learning guarantees. *Mach. Learn. 73*, 3 (2008), 273–287.

[2] Ben-david, S., and Schuller, R. Exploiting task relatedness for multiple task learning. In *In Proceedings of the sixteenth annual conference on learning theory (COLT)* (2003).

[3] Caruana, R. Multitask learning. In *Machine Learning* (1997), pp. 41–75.

[4] Daumé III, H. Bayesian multitask learning with latent hierarchies. In *Conference on Uncertainty in Artificial Intelligence* (Montreal, Canada, 2009).

[5] Jacob, L., Bach, F., and Vert, J.-P. Clustered multi-task learning: A convex formulation. In *NIPS* (2008).

[6] Micchelli, C. A., and Pontil, M. Regularized multi-task learning. In *KDD* (2004).

[7] Thrun, S., and Pratt, L., Eds. *Learning to learn.* Kluwer Academic Publishers, Norwell, MA, USA, 1998.