Knowl Inf Syst (2016) 49:933–973
DOI 10.1007/s10115-016-0926-z

REGULAR PAPER

A new transfer learning framework with application to model-agnostic multi-task learning

Sunil Gupta[1] · **Santu Rana**[1] · **Budhaditya Saha**[1] · **Dinh Phung**[1] · **Svetha Venkatesh**[1]

Received: 17 November 2014 / Revised: 22 November 2015 / Accepted: 3 February 2016 /
Published online: 19 February 2016

Abstract Learning from small number of examples is a challenging problem in machine learning. An effective way to improve the performance is through exploiting knowledge from other related tasks. Multi-task learning (MTL) is one such useful paradigm that aims to improve the performance through jointly modeling multiple related tasks. Although there exist numerous classification or regression models in machine learning literature, most of the MTL models are built around ridge or logistic regression. There exist some limited works, which propose multi-task extension of techniques such as support vector machine, Gaussian processes. However, all these MTL models are tied to specific classification or regression algorithms and there is no single MTL algorithm that can be used at a meta level for any given learning algorithm. Addressing this problem, we propose a *generic, model-agnostic* joint modeling framework that can take any classification or regression algorithm of a practitioner's choice (standard or custom-built) and build its MTL variant. The key observation that drives our framework is that due to small number of examples, the estimates of task parameters are usually poor, and we show that this leads to an under-estimation of task relatedness between any two tasks with high probability. We derive an algorithm that brings the tasks closer to their true relatedness by improving the estimates of task parameters. This is achieved by appropriate sharing of data across tasks. We provide the detail theoretical underpinning of the algorithm. Through our experiments with both synthetic and real datasets, we demonstrate that the multi-

✉ Sunil Gupta
sunil.gupta@deakin.edu.au

Santu Rana
santu.rana@deakin.edu.au

Budhaditya Saha
budhaditya.saha@deakin.edu.au

Dinh Phung
dinh.phung@deakin.edu.au

Svetha Venkatesh
svetha.venkatesh@deakin.edu.au

[1] Center for Pattern Recognition and Data Analytics (PRaDA), Deakin University, Geelong Waurn Ponds Campus, Waurn Ponds, VIC, Australia

task variants of several classifiers/regressors (logistic regression, support vector machine, K-nearest neighbor, Random Forest, ridge regression, support vector regression) convincingly outperform their single-task counterparts. We also show that the proposed model performs comparable or better than many state-of-the-art MTL and transfer learning baselines.

Keywords Multi-task learning · Model-agnostic framework · Meta algorithm · Classification · Regression

1 Introduction

Machine learning field is rich in supervised algorithms to perform regression and classification. The algorithms take data with labels or response as input and generate a predictive model as output. To capture complex patterns of real-world phenomenon, these algorithms usually require a large amount of data. However, in reality, to the dislike of machine learning practitioners in many situations, only a limited amount of data is available to learn from. Learning from small amount of data often results in models that have poor performance on the unseen data. For example, the amount of email/spam data of a new user is often insufficient to build a personalized spam detection model with acceptable utility. Similarly, survival prediction for a rare disease is often a difficult task due to the lack of availability of sufficient data. In such situations, an effective way to improve the performance or generalization ability of a model is through exploiting knowledge from other related tasks, e.g., exploiting spam detection data of other users or survival data of other related common diseases. Multi-task learning (MTL) [10] is one such useful paradigm. A set of seemingly related tasks are learnt together to mitigate the effect of insufficiency of data in individual tasks.

A naïve approach to MTL is to combine data from all the tasks and learn a single model. However, this approach does not work when some of the tasks are partially related or unrelated needing separate models for them. Instead, a more sensible approach is to combine them in proportion of their relatedness. Realizing this point, most of the MTL techniques combine tasks in a more flexible manner, e.g., learning a representation of model parameters in a subspace where related tasks share the same basis vectors, while unrelated tasks are allowed to have their own bases [2,32,33]; imposing a common prior distribution on task parameters [15,36,60]; or a combination of both these approaches [27,46]. Taking an explicit approach, Zhang et al. [63] directly learn task relationship via a covariance matrix. Thus building a MTL algorithm is a higher level concept that involves a way of learning task relatedness and combining the tasks accordingly.

Although there exist numerous classification or regression models in the machine learning literature, most of the MTL models are built around ridge or logistic regression [2,20,33]. There exist limited works, which propose multi-task extension of techniques such as support vector machine (SVM), decision trees, Gaussian processes etc [6,55,60,65]. However, all of MTL models are tied to a specific classification or regression technique. It is well known in the machine learning community that no single classification or regression technique is the best for all applications. For example, logistic regression is popular in clinical applications due to its feature interpretability [54,64]. Similarly, Random Forest is a popular model for its ease of use [9]. For text classification, naïve Bayes remains a popular classification model [43,47] and when one does not want to train any model, K-nearest neighbor (KNN) may be a way to go. Unfortunately, the current MTL algorithms are developed such that when changing the base classifier or regressor technique, one is also *required* to change the MTL

algorithm. Since MTL is a higher level concept that is independent of a classifier/regressor, it may be possible to untie it from the underlying techniques of classification or regression. In that way, MTL framework can be enabled for any standard, custom-built or even yet to be discovered classification /regression techniques.

Motivated by the need of a *generic MTL* algorithm, we propose a transfer learning framework that can take any classification or regression technique of a practitioner's choice (standard or custom-built) and build its MTL variant. We call this framework *Multi-Task Transfer Learning* (MTTL). To this end, we introduce *a novel task relatedness measure* that can be computed in a way agnostic to the classification or regression technique being used. Tasks are then combined in a novel way according to their relatedness. As a theoretical basis of our framework, we show that under some mild assumptions, and in the regimen of small training data, the estimates of task parameters are usually poor leading to under-estimation of task relatedness with high probability. Following that, we derive an algorithm that brings the tasks closer to their "true" relatedness. This process indirectly leads to improved estimates of task parameters. At the core, the process operates by appropriate sharing of data across tasks and thus agnostic to the choice of base classification or regression technique. We provide MTTL extensions for many popular techniques namely, logistic regression, SVM, KNN, Random Forest, ridge regression, support vector regression. Through our experiments with both synthetic and real datasets, we demonstrate that the multi-task variants of all these techniques convincingly outperform their single-task counterparts. We also show that the proposed model performs comparable or better than many state-of-the-art MTL and transfer learning baselines.

Our main contributions are:

- We propose a generic transfer learning framework that can take any classification or regression algorithm of a practitioner's choice (standard or custom-built) and build its MTL variant.
- We introduce a new task relatedness measure that is agnostic to a specific classifier or regression model.
- Using synthetic data, we illustrate the behavior of the proposed MTTL algorithm on two important aspects: (1) its behavior under varying degree of task relatedness (2) its behavior when data are nonlinearly separable. Using four real datasets (two classification and two regression datasets), we demonstrate the usefulness of the proposed model for various real-world applications.

We favor a unified view of transfer learning and multi-task learning. Both learning paradigms aim to transfer knowledge from one task to other tasks. Transfer learning is seen as an umbrella term, while MTL is usually used in more specific sense when there are multiple tasks and the tasks are learnt simultaneously to mutually transfer knowledge to each other. In the setting considered in this paper, we have *multiple tasks* and our *goal* is to mutually transfer knowledge across all these tasks. However, our work differs from the traditional MTL methods in the manner this goal is achieved—the learning is not simultaneous. This has been necessitated due to the need of model-agnostic knowledge transfer. Since for many models such as Random Forest there is no parametric form of classifier, it is not possible to learn a common representation across tasks and use that for simultaneous learning. *Therefore, we have used the idea of transfer learning to achieve what is usually done by MTL.* Since we are trying to achieve MTL via transfer learning (through sharing data among tasks), we call our framework as "multi-task transfer learning".

2 Background

In this section, we provide a brief review of some of the existing techniques in the area of MTL and transfer learning.

2.1 Multi-task learning (MTL) methods

The MTL is a learning paradigm in which multiple related tasks are learned jointly in order to achieve better performance for each individual task. This strategy is especially beneficial when individual tasks have small number of instances and the performance of a model learnt from a single task is poor. The main idea is to leverage the common knowledge present across tasks toward improving mutual performance. Early works in this area naïvely combine tasks for joint learning [19,20]. However, when some of the tasks are unrelated or negatively related, joint learning with them may be detrimental to performance. To address this issue, later works on MTL estimate some form of *task relatedness* and use it for combining the tasks in a more flexible manner. For example, subspace-based models that learn a representation of model parameters in a lower dimensional subspace where related tasks share the same basis vectors, while unrelated tasks are allowed to have their own bases [2,32,33]. Argyriou et al. [2] propose a model that represents task parameters in a single low-dimensional subspace and imposes a mixed norm (ℓ_2/ℓ_1) penalty on the representation matrix. This is performed to obtain a sparse representation, which ensures that a task parameter uses only a subset of the basis vectors in the learnt subspace and thus only those tasks that share common bases are combined together for MTL. Although this is an effective way of doing MTL, the use of a single subspace that is learnt from all the tasks data may still cause problems due to the presence of unrelated tasks, leading to performance degradations. Gong et al. [24] propose a decomposition of the parameter subspace for explicitly separating outlier tasks. Taking a more general approach, Kang et al. [32] propose a framework that puts the related tasks into groups while learning a subspace for each group. Similar attempts have been made by Kumar et al. [33], who use l_1 norm instead of using mixed-norm penalty. [27,46]. Some of the probabilistic approaches toward MTL use a common prior distribution on task parameters [15,36,60]. The main idea is that using a prior distribution keeps each task flexible enough while still sharing the common knowledge across them by using a single-probability distribution-based generative model. Again, when the tasks are partially related, unrelated and even negatively related, these techniques may not be very effective. Some of the recent techniques that combine the probabilistic approach with subspace-based approach are proposed in [27,46]. These techniques are based on Bayesian nonparametric modeling and are capable of inferring the model parameters automatically from the data. However, model inference of Bayesian techniques is known to be notoriously slow and often does not scale well. To model task relatedness, a relatively simpler approach is taken by Zhang et al. [63], who use a covariance matrix to learn task relationship and combine the task parameters accordingly. Other recent works that attempt to capture locally transferrable patterns have taken different approaches, e.g. [29] learn multiple facets of task relationships and transfer knowledge differently for each facet. A similar idea where multiple relations are used for differential knowledge transfer is proposed in [28]. To handle scenario where the number of samples are extremely small, Saha et al. [51] have proposed a MTL approach based on data-sharing approach. A theoretical analysis of MTL and its benefits are discussed in [3].

Most of the MTL models are developed to be used with a particular classification or regression technique. Some of these models that are applied for *classification* are: Logistic regression [33], SVM [65], Gaussian processes [8,60] and Decision tree [55] etc. The other

examples that are applied for *regression* are: Ridge regression, Kernel ridge regression [2,19] etc. There is no MTL model in the literature that can be used in combination with a variety of classification or regression models. This problem necessitates the need to develop a generic MTL algorithm, which is our main goal in this paper.

2.2 Transfer learning methods

A closely related concept to MTL is Transfer Learning. Most of the MTL techniques aim to learn a task parameter as a function of data and other task parameters. In contrast, transfer learning techniques focus on a 'target' task and learn from data of other 'source' tasks. The parameter estimation for each task is separately performed. Transfer learning has its roots in the concept of 'learning to learn' where the goal is to use knowledge acquired in the past while learning a new concept [53]. Applications of transfer learning involves semi-supervised learning or learning from limited examples in the target task given plentiful examples in the source task. Knowledge acquired from the source task is uni-directionally transferred to the target task. Transfer learning algorithms support cross-domain knowledge transfer [11,41], even when no labeled examples are available in target tasks [50,56]. Learning techniques generally involve transferring knowledge of instances such as *TrAdaBoost* [14] and others [5,13,40], or transferring knowledge of feature representation, both supervised [18,31,36] and unsupervised [35,50], or transferring knowledge of model parameters [8,17,22,34,57], or transferring relational knowledge [44]. In literature, transfer learning techniques have appeared under different names—e.g., 'Inductive transfer learning', 'Transductive transfer learning' or 'Unsupervised transfer learning'—often, due to their suitability under slightly different application settings. The term 'Inductive transfer learning' has been used to refer to problems where target data have supervision information, e.g., MTL [2] and self-taught learning [50]. Similarly, 'transductive transfer learning' is used to refer to problems where target data either has very limited supervision information or no information at all, e.g., domain adaptation [16], sample selection bias [61] and co-variate shift [52]. Finally, the term 'Unsupervised transfer learning' has been used to refer to problems where both source and target tasks have no supervision information, e.g., shared subspace learning [25,26]. A more comprehensive survey of transfer learning is available in [45]. For a theoretical discussion on the benefits of the transfer learning, we refer the reader to [4] and [48].

3 The sample selection multi-task transfer learning

We propose a framework that allows us to apply the idea of MTL at a meta level with any supervised model, e.g., logistic/ridge regression, SVMs, naïve Bayes, KNN, Random Forest etc. Typically a MTL framework jointly models related tasks by combining them according to their task relatedness. This framework has two elements—joint modeling and computation of task relatedness. Both elements usually require an *explicit* representation of model parameters, which hinders the construction of MTL models in a generic setting, e.g., it is not clear how to compute task relatedness between two tasks or do their joint modeling when these tasks are using Random Forest or KNN classifiers. To address this problem, we first introduce a novel *task relatedness measure* that is agnostic to the choice of underlying classifier or regression model. Next, we carry out joint modeling of multiple tasks by appropriately *sharing data* among them based on their task relatedness. Joint modeling via data sharing allows us to use any underlying classification or regression model even in absence of an explicit representation of model parameters. In the following, we first define

appropriate notations and provide a brief sketch of our framework before going into the details from Sect. 3.1 onwards.

Consider a domain $\mathcal{D} = (\mathcal{X}, \mathcal{Y}, \mathcal{P})$, where $\mathcal{X}$ denotes a feature space, $\mathcal{Y}$ denotes a outcome space and $\mathcal{P}$ is a set of probability distributions over $(\mathcal{X}, \mathcal{Y})$. Given domain $\mathcal{D}$, data of task t, i.e. $(\mathbf{X}_t, \mathbf{y}_t) = \left\{\left(\mathbf{x}_1^t, y_1^t\right), \ldots, \left(\mathbf{x}_{N_t}^t, y_{N_t}^t\right)\right\}$ is drawn independently according to some underlying distribution P_t on $\mathcal{X} \times \mathcal{Y}$. Assuming that we have T tasks, multi-task data can be summarized as the collection of $\{(\mathbf{X}_t, \mathbf{y}_t)\}_{t=1}^T$. We use M to denote the number of features and N_t to denote the number of instances in task t. In the standard *single-task learning* (STL), the parameter of task t, denoted as $\mathcal{M}_t$, is learnt independently from its data $(\mathbf{X}_t, \mathbf{y}_t)$ using a supervised learning model, i.e.

$$STL{:}\mathcal{M}_t \leftarrow \text{supervised learner } (\mathbf{X}_t, \mathbf{y}_t), \quad \text{for } t = 1, \ldots, T \tag{1}$$

Traditionally, MTL models multiple related tasks jointly and estimates the task parameter of task t (i.e. $\mathcal{M}_t$) as a function of some statistics derived from task t's data, i.e. $(\mathbf{X}_t, \mathbf{y}_t)$ and the task parameters of other tasks (i.e. $\mathcal{M}_{t'}, \forall t' \neq t$). We refer to this approach as "parameter-sharing" approach. We note that this way of doing MTL requires exact parametrization of $\mathcal{M}$ in a compact form so that $\mathcal{M}_t$ can be learnt as a function of $\mathcal{M}_{t'}, \forall t' \neq t$. This brings restrictions when extending these MTL ideas for various supervised base models (classifiers/regressors) where it is hard to represent the base models through a compact parametrization. For example, it is not clear how to do MTL taking a parameter-sharing approach when a K-nearest neighbor (KNN) or Random Forest or naïve Bayes is used as base classifier. Although it may be possible to deal with some of the base classifiers on an *ad hoc* basis, this way of doing MTL is usually hard [7,8,63,65]. We address this limitation by using "data-sharing" approach in contrast to the "parameter-sharing" approach of the traditional MTL algorithms. Some of the characteristics of our proposed framework are as follows:

- Our framework imposes the commonality across the tasks by sharing data instead of sharing parameters. Sharing the training data with other tasks brings them closer. When tasks are negatively correlated, sharing their data with labels flipped (or targets reversed) takes them away.
- A key advantage of our proposed data-sharing approach is that it allows us to develop MTL idea on any base supervised model (*generic framework*).

In our proposed multi-task transfer learning (MTTL), a task borrows data from other tasks in a probabilistic manner. The need of borrowing data is justified in later part of this section using a *crucial* result (Theorem 1). Further results (Proposition 1 and Theorem 2) help derive a probability and an algorithm for borrowing data that ensures that tasks are brought closer together or pushed farther apart according to their relatedness.

An overall sketch of our proposed MTTL is as follows. Let us assume that task t borrows data from task t' (indexed as $n = 1, \ldots, N_{t'}$) using probabilities $\{p_{tn}\}_{n=1}^{N_{t'}}$ where $t' \neq t$. To ensure that data from other tasks participate in the learning of the model of task t (i.e. $\mathcal{M}_t$) with these probabilities, we create an *ensemble* of models. Let us assume that there are R models in this ensemble, indexed as $r = 1, \ldots, R$. The r-th model in the ensemble $\mathcal{M}_t^{(r)}$ is trained using the data $\left(\left[\mathbf{X}_t, \mathbf{X}_t^{-t,r}\right], \left[\mathbf{y}_t, \mathbf{y}_t^{-t,r}\right]\right)$ where we denote the data from other tasks that are borrowed by the t-th task in r-th random draw by a matrix $\mathbf{X}_t^{-t,r}$ and the corresponding labels/targets by a vector $\mathbf{y}_t^{-t,r}$. Formally, our MTTL framework learns the r-th model in the ensemble as

$$MTTL: \mathcal{M}_t^{(r)} \leftarrow \text{supervised learner } \left(\left[\mathbf{X}_t, \mathbf{X}_t^{-t,r}\right], \left[\mathbf{y}_t, \mathbf{y}_t^{-t,r}\right]\right), \quad \text{for } t = 1, \ldots, T \tag{2}$$

Using the ensemble of R models, prediction for a new data $\mathbf{x}^t_{\text{new}}$ is made as

$$\mathcal{M}_t\left(\mathbf{x}^t_{\text{new}}\right) = \frac{1}{R}\sum_{r=1}^{R}\mathcal{M}_t^{(r)}\left(\mathbf{x}^t_{\text{new}}\right) \tag{3}$$

It is immediately clear from (2) and (3) that our proposed MTTL framework can be used with any supervised model as the only change from STL to MTTL case is the new augmented training set. We refer to our proposed framework as sample selection Multi-Task Transfer Learning (ssMTTL).

In the remainder of this section, we describe why joint modeling based on borrowing data makes sense and how to borrow "useful" data from other tasks.

3.1 Why joint modeling based on borrowing data makes sense?

In this section, we first define a measure of task relatedness and then present a result in Theorem 1 which motivates the need of borrowing data.

Definition 1 (*Task relatedness*) The degree of relatedness of task t with the task t' (denoted as $\rho_{t,t'}$) is *defined* as the level of agreement between the predictions made by the two tasks on the training data of task t'.

The above task relatedness is computed as follows. Let us assume, on the training data of task t', predictions made by task t and t' are represented as vectors $\mathbf{v}_t$ and $\mathbf{v}_{t'}$. Now, the task relatedness $\rho_{t,t'}$ is defined as a *correlation* between the vectors $\mathbf{v}_t$ and $\mathbf{v}_{t'}$. In practice, $\mathbf{v}_t$ and $\mathbf{v}_{t'}$ can be predicted labels or scores in case of classification tasks, or predicted target values in case of regression tasks. A robust option in case of classification problems with imbalanced classes is the ranking over the scores, i.e. using Spearman correlation instead of Pearson correlation.

Conventionally, task relatedness is directly measured as the correlation between task parameters when explicit parametrization is available. Although our definition seems different from the conventional one, it converges to the same relatedness measure when there are sufficiently large samples in task t'. Additionally, it can be used with any underlying classification or regression models as it does not need explicit parametrization.

Using the above task relatedness, the following theorem sets the case for MTTL, in particular, it states what happens when tasks are learnt independently and motivates the need for joint modeling to achieve improved learning.

Theorem 1 *Under the assumption of a symmetric error distribution for task-relatedness estimates, the probability of under-estimation of task relatedness between any two tasks that are positively related, is higher than that of over-estimation. Converse is true for negatively related tasks.*

Proof Consider any two tasks t and t' and denote their predictions made on the training data of task t', as vectors v_t and $v_{t'}$. Given that there are $N_{t'}$ instances in task t', the vectors $\mathbf{v}_t$, $\mathbf{v}_{t'}$ are of length $N_{t'}$ but the underlying space where they lie may be smaller. Let d be the dimension of this underlying space. [1] Following Definition 1, let us denote the *true* and *estimated* task relatedness between these tasks by $\rho_{t,t'}$ and $\hat{\rho}_{t,t'}$ respectively. Let α be an angle such that $\alpha \triangleq \cos^{-1}\left(\rho_{t,t'}\right)$ and $\hat{\alpha}$ be the angle measured between the vectors $\mathbf{v}_t$ and $\mathbf{v}_{t'}$ and defined as $\hat{\alpha} \triangleq \cos^{-1}\left(\hat{\rho}_{t,t'}\right)$. We note that $\hat{\alpha}$ is an unbiased estimate of α, however, as it is

[1] In case of a general nonlinear model, $d \leq N_{t'}$. For linear models, assuming a linearly independent set of data in task t', $d = \min\left(M, N_{t'}\right)$.

derived from a finite number of instances, it has a distribution around α. We assume a uniform distribution for the estimation error, i.e. $\left(\hat{\alpha} - \alpha\right) \sim \mathcal{U}\left[-\phi, \phi\right]$ where ϕ dictates the variance of the distribution (var $\left(\hat{\alpha} - \alpha\right) = \frac{\phi^2}{3}$) and inversely related to the number of instances in task t and t', i.e. N_t and $N_{t'}$. Figure 1 provides an illustration of under-estimation/over-estimation of task relatedness under two different error distribution assumptions: uniform distribution (relevant plots are Fig. 1a–c) and *von-Mises* error distribution (relevant plots are Fig. 1d, e).

Figure 1c depicts vectors $\mathbf{v}_t$, $\mathbf{v}_{t'}$ in a three-dimensional space and marks the regions of under-estimation and over-estimation using two cones in 'blue' and 'orange' color respectively. Without loss of generality, the vector $\mathbf{v}_t$ is shown to lie along the axis of the first cone and given a task relatedness $\rho_{t,t'}$ (or equivalently α), $\mathbf{v}_{t'}$ can then be a ray starting from the apex (or origin) and lying on the surface of this cone. The second cone (shown in 'orange' color) in Fig. 1c depicts the extent of uniform distribution of the error in task relatedness. We note that, for any given $\mathbf{v}_{t'}$, this cone is formed by using $v_{t'}$ as axis of the cone and has an aperture of 2ϕ around it axis. This distribution has a direct role in computing both under-estimation and over-estimation probabilities. Clearly, for a fixed $\mathbf{v}_t$, an estimated $\mathbf{v}_{t'}$ that lies within the aperture of the first cone is over-estimated as it now closer to $\mathbf{v}_t$ compared to the true $\mathbf{v}_{t'}$. The probability of this *over-estimation* (denoted as $\mathcal{P}\left(\hat{\rho}_{t,t'} > \rho_{t,t'}\right)$) can thus be computed by measuring the intersection of solid angles formed by the first and the second cone and is given by

$$\mathcal{P}\left(\hat{\rho}_{t,t'} > \rho_{t,t'}\right) \propto \begin{cases} \frac{1}{2} I_{\sin^2 \frac{\phi}{2}}\left(\frac{d-1}{2}, \frac{1}{2}\right) & \text{if } \phi \leq 2\alpha \\ \frac{1}{2} I_{\sin^2 \alpha}\left(\frac{d-1}{2}, \frac{1}{2}\right) & \text{if } \phi > 2\alpha \end{cases} \tag{4}$$

where $I_z\left(a, b\right)$ is *regularized* Beta function and defined as $I_z\left(a, b\right) = \frac{B(z;a,b)}{B(a,b)}$ and $B\left(z; a, b\right)$ is *incomplete* Beta function. For details on the closed form formula of solid angle, we refer the reader to [39]. In the above, without loss of generality, we assume the tasks to be positively related, i.e. $\alpha \leq \pi/2$ and $\phi \leq \pi$. When the tasks are negatively related, the proof follows by replacing α with $\pi - \alpha$.

The *under-estimation* region is formed by all rays in the second cone that do not belong to the first cone. Therefore, the probability of under-estimation (denoted as $\mathcal{P}\left(\hat{\rho}_{t,t'} \leq \rho_{t,t'}\right)$) can be written as

$$\mathcal{P}\left(\hat{\rho}_{t,t'} \leq \rho_{t,t'}\right) \propto \begin{cases} \frac{1}{2} I_{\sin^2 \frac{\phi}{2}}\left(\frac{d-1}{2}, \frac{1}{2}\right) & \text{if } \phi \leq 2\alpha \\ \frac{1}{2} I_{\sin^2 (\phi - \alpha)}\left(\frac{d-1}{2}, \frac{1}{2}\right) & \text{if } \phi > 2\alpha \end{cases} \tag{5}$$

To prove the theorem, we need to show that $\mathcal{P}\left(\hat{\rho}_{t,t'} \leq \rho_{t,t'}\right) \geq \mathcal{P}\left(\hat{\rho}_{t,t'} > \rho_{t,t'}\right)$. The equality $\mathcal{P}\left(\hat{\rho}_{t,t'} > \rho_{t,t'}\right) = \mathcal{P}\left(\hat{\rho}_{t,t'} \leq \rho_{t,t'}\right)$ is clearly observed when $\phi \leq 2\alpha$. For the case, when $\phi > 2\alpha$, using (4) and (5), the probability of over-estimation $\mathcal{P}\left(\hat{\rho}_{t,t'} > \rho_{t,t'}\right)$ can be re-written as

$$\mathcal{P}\left(\hat{\rho}_{t,t'} > \rho_{t,t'}\right) = \frac{I_{\sin^2 \alpha}\left(\frac{d-1}{2}, \frac{1}{2}\right)}{I_{\sin^2 \alpha}\left(\frac{d-1}{2}, \frac{1}{2}\right) + I_{\sin^2 (\phi - \alpha)}\left(\frac{d-1}{2}, \frac{1}{2}\right)}$$

Since $\phi > 2\alpha$, we have

$$\sin^2\left(\phi - \alpha\right) > \sin^2 \alpha \implies I_{\sin^2 (\phi - \alpha)}\left(\frac{d-1}{2}, \frac{1}{2}\right) > I_{\sin^2 \alpha}\left(\frac{d-1}{2}, \frac{1}{2}\right)$$

Using above, we have $\mathcal{P}\left(\hat{\rho}_{t,t'} > \rho_{t,t'}\right) < \frac{1}{2}$. Since $\mathcal{P}\left(\hat{\rho}_{t,t'} > \rho_{t,t'}\right) + \mathcal{P}\left(\hat{\rho}_{t,t'} \leq \rho_{t,t'}\right) = 1$, we also have $\mathcal{P}\left(\hat{\rho}_{t,t'} \leq \rho_{t,t'}\right) \geq \frac{1}{2}$. Therefore, we have $\mathcal{P}\left(\hat{\rho}_{t,t'} \leq \rho_{t,t'}\right) \geq \mathcal{P}\left(\hat{\rho}_{t,t'} > \rho_{t,t'}\right)$.

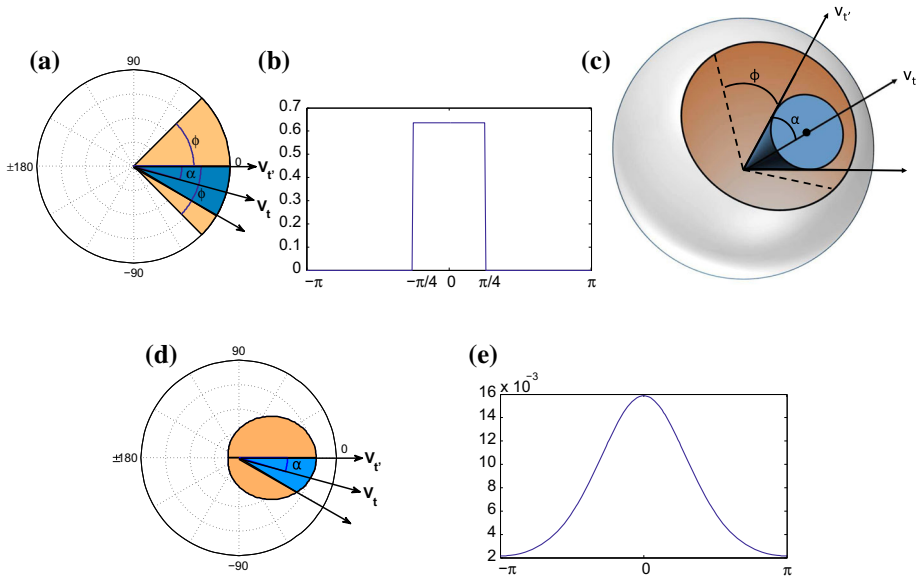

Fig. 1 An illustration of under-estimation of task relatedness: **a** a polar plot depicting the over-estimation area in '*blue*' color and under-estimation area in '*orange*' color under the assumption of *uniform* error distribution (between [-ϕ,ϕ]) for task relatedness, **b** Cartesian representation of the uniform error distribution, **c** a similar plot depicting the over-estimation and the under-estimation in a three-dimensional space, **d** a polar plot depicting the over-estimation area in '*blue*' color and under-estimation area in '*orange*' color under the assumption of *von-Mises* error distribution, **e** Cartesian representation of the *von-Mises* distribution. In our task relatedness definition (see Definition 1) of $\rho_{t,t'}$, the lower the number of examples in tasks t and t', the higher is the variance of the error distribution and thus higher is the probability of under-estimation due to the fatter distribution-tails

When the tasks are negatively related, i.e. when $\alpha > \pi/2$, this inequality relation changes to $\mathcal{P}\left(-\hat{\rho}_{t,t'} \leq -\rho_{t,t'}\right) \geq \mathcal{P}\left(-\hat{\rho}_{t,t'} > -\rho_{t,t'}\right)$. Therefore, in general, we have

$$\mathcal{P}\left(|\hat{\rho}_{t,t'}| \leq |\rho_{t,t'}|\right) \geq \mathcal{P}\left(|\hat{\rho}_{t,t'}| > |\rho_{t,t'}|\right) \tag{6}$$

We note that the above proof, to keep the matter simple, assumes a *uniform* distribution for the error in task relatedness estimation, however, the statement of Theorem 1 is valid for any symmetric error distribution, e.g., *von-Mises-Fisher* distribution [42]. We note that the difference in under-estimation and over-estimation probability depends on the number of examples available in the task t and t', which directly affects the variance of the error distribution. Lower the number of examples in the task t, higher is the variance of the error distribution and therefore higher is the under-estimation probability due to a fatter distribution tail. □

Remark 1 It follows from Theorem 1 that $\mathcal{P}\left(|\hat{\rho}_{t,t'}| \leq |\rho_{t,t'}|\right) \geq \mathcal{P}\left(|\hat{\rho}_{t,t'}| > |\rho_{t,t'}|\right)$. In fact, the *equality* is satisfied only when tasks are unrelated ($\alpha = 90°$) *or* when the variance of the error distribution is low enough to contain the over-estimation region, i.e. $\phi \leq 2\alpha$. An illustration of the result established by Theorem 1 (simulated for the case of uniform error distribution with $\phi = 180°$) is provided in Fig. 2. As seen from the figure, the probability of the under-estimation of task relatedness increases with the true correlation. This increasing behavior becomes even more pronounced in higher dimensional spaces, e.g., in

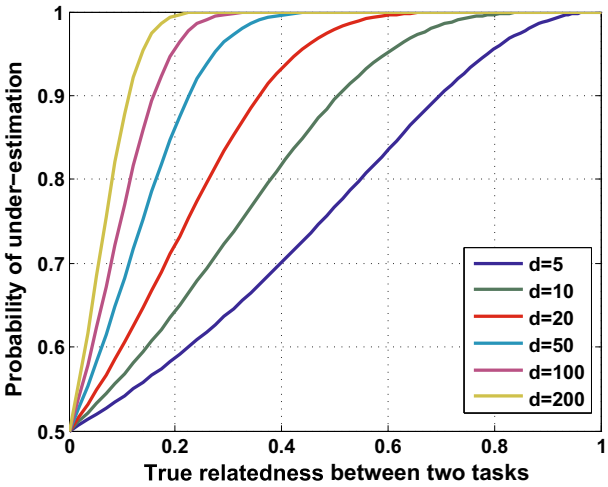

Fig. 2 An illustration of probability of under-estimation for task relatedness with varying degrees of true relatedness and dimensions (d). For this plot, the angular error in task relatedness estimation is assumed to be uniformly distributed, i.e. $\left(\hat{\alpha} - \alpha\right) \sim \mathcal{U}\left[-\phi, \phi\right]$ with $\phi = 180°$

a 100-dimensional space, the probability of under-estimation becomes higher than 0.95 as soon as the true correlation exceeds merely a small value of 0.2. In MTL applications, since the tasks usually have fairly high correlations, we may safely assume that task relatedness estimated by single-task learning are under-estimated with a high probability. Assuming $M < N_{t'}$, the dimension d is the dimension of the feature space (M) for linear models. For nonlinear models, d is usually more than M as a nonlinear function can often be thought of a linear function in a higher dimensional feature space, i.e. we have $M \leq d \leq N_{t'}$. *Therefore, the difference between the under-estimation and the over-estimation probabilities is more pronounced for nonlinear models than the linear models for the same problem. This makes MTL even more relevant for nonlinear modeling.*

Remark 2 Another important aspect of the above result, which is relevant to MTL is that when the number of examples in tasks t and t' increase, the variance of the error distribution (angular error in task relatedness estimation) starts to decrease. Depending on how related the tasks are when the variance decreases below a certain threshold, the probability of under-estimation and over-estimation become almost equal. From this point onwards, making tasks similar via MTL may not lead to any significant improvements in parameter estimation. For tasks which are less related the threshold is higher than that for tasks which are highly related. A clear instance of this point can be seen for the uniform error distribution (i.e. $\mathcal{U}\left[-\phi, \phi\right]$). When N_t and $N_{t'}$ become sufficiently large and ϕ decreases to a value such that $\phi = 2\alpha$ then the probability of under-estimation becomes equal to that of the over-estimation. Since, for highly related tasks α is smaller than the less related tasks, the ϕ that makes the probabilities equal is higher for less related tasks than the highly related tasks. *In essence, MTL is beneficial when either tasks are small or tasks are highly related.*

From Theorem 1, it is clear that for any two related tasks, there is a higher chance that task relatedness *estimated* using STL is lower than the *true* task relatedness. We can remedy this by taking some data points of one task and including them into the dataset of the other task to alter their relatedness. *However, data borrowing (or sample selection) process is non-trivial*

and should be done carefully so that relatedness between the two tasks reaches just up to their true relatedness. In the next subsection, we present Theorem 2, which helps deriving an algorithm that guarantees only *appropriate shift* in the task relatedness.

3.2 How to select "useful" data from other tasks

Before we proceed into the details for the data selection process, we provide here some of the terminologies to be used in the subsequent discussion.

3.2.1 Data Likelihood model and related terms

– For classification tasks, the *likelihood* of an instance $(\mathbf{x}_n, y_n)$ w.r.t. a classifier $\mathcal{M}_t$ is denoted as $l(\mathbf{x}_n, y_n; \mathcal{M}_t)$ and is defined as

$$l(\mathbf{x}_n, y_n; \mathcal{M}_t) = (s(\mathbf{x}_n, \mathcal{M}_t))^{y_n} (1 - s(\mathbf{x}_n, \mathcal{M}_t))^{(1-y_n)} \tag{7}$$

where $s(\mathbf{x}_n, \mathcal{M}_t)$ is a predicted score associated to $\mathbf{x}_n$ by classifier $\mathcal{M}_t$ and lies between 0 and 1. The score function $s(\mathbf{x}_n, \mathcal{M}_t)$ can take different forms for different classifiers. For example, in the case of logistic regression, it uses a logistic sigmoid function on the output $\mathcal{M}_t(\mathbf{x}_n)$, i.e. $s(\mathbf{x}_n, \mathcal{M}_t) = \frac{1}{1+\exp(-\mathcal{M}_t(\mathbf{x}_n))}$. For the support vector classification, $s(\mathbf{x}_n, \mathcal{M}_t)$ is computed as calibrated posterior probability using Platt's sigmoid fitting [49]. Similarly, for the Naïve Bayes, it is computed as the posterior probability $P(y_n = 1 \mid \mathbf{x}_n)$ and for the Random Forest, it is computed by averaging the predicted labels (0 or 1) of all the trees.

– For regression tasks, the *likelihood* of data $(\mathbf{x}_n, y_n)$ w.r.t. the function $\mathcal{M}_t$ is denoted as $l(\mathbf{x}_n, y_n; \mathcal{M}_t)$ and is defined as

$$l(\mathbf{x}_n, y_n; \mathcal{M}_t) = \exp(- \mid y_n - \mathcal{M}_t(\mathbf{x}_n) \mid /\sigma) \tag{8}$$

where σ is the scale parameter.

– The *novelty* of data $(\mathbf{x}_n, y_n)$ to the classifier $\mathcal{M}_t$ is defined as follows

$$h(\mathbf{x}_n, y_n; \mathcal{M}_t) = 1 - l(\mathbf{x}_n, y_n \mid \mathcal{M}_t) \tag{9}$$

This measure is used to compute how novel a classifier $\mathcal{M}_t$ finds a data point $(\mathbf{x}_n, y_n)$ of some other task t'.

– The *normality* of data $(\mathbf{x}_n, y_n)$ is its likelihood w.r.t. its own task classifier $\mathcal{M}_t$.

3.2.2 Data selection scheme

Our framework selects any data instance from other tasks *probabilistically*. To ensure that a data instance participates in the model with *a certain probability*, we create an ensemble of models. The r-th model $(\mathcal{M}_t^{(r)})$ in the ensemble for the task t is trained using the data $\left[\mathbf{X}_t, \mathbf{X}_t^{-t,r}\right]$ as in (2) where the additional data $\mathbf{X}_t^{-t,r}$ is constructed by uniformly randomly sampling data from other tasks with probabilities of (10). Prediction is made by averaging the output of each model in the ensemble using (3). The following proposition details a scheme for data or sample selection, which ensures that tasks are brought closer together or pushed farther apart to bring them closer to their true relatedness.

Proposition 1 *The probability of including n-th data point of task t' for a target task t is given as*

$$p_{tn} = \underbrace{q_{t,t'}}_{\text{sampling prob.}} \times \underbrace{h\left(\mathbf{x}_n, y_n \mid \mathcal{M}_t\right)}_{\text{novelty}} \times \underbrace{l\left(\mathbf{x}_n, y_n \mid \mathcal{M}_{t'(n)}\right)}_{\text{normality}} \tag{10}$$

where $q_{t,t'}$ is the probability of sampling data that brings the tasks up to their true relatedness. The term $h\left(\mathbf{x}_n, y_n \mid \mathcal{M}_t\right)$ ensures that novel data points from task t' are selected with higher probability. The term $l\left(\mathbf{x}_n, y_n \mid \mathcal{M}_{t'(n)}\right)$ measures the normality of the data point $(\mathbf{x}_n, y_n)$, which *guards against the outlier points* from task t' being included. The sampling probability $q_{t,t'}$ is computed as

$$q_{t,t'} = \frac{\eta \hat{\rho}_{t,t'}}{1 - \hat{\rho}_{t,t'}} \tag{11}$$

where $\rho_{t,t'}$ and $\hat{\rho}_{t,t'}$ are the true and the estimated relatedness between the tasks t and $t'\,(n)$, and η is a parameter. The value of η directly controls the level of sharing between tasks and can be learnt through model selection.

Theorem 2 *Sampling data with probability in* Eq. (11) *brings the under-estimated task relatedness to its true value with high probability.*

Proof We provide the proof for classification problems. However, similar arguments also hold for regression problems. Given two tasks t and t', estimated task relatedness $\hat{\rho}_{t,t'}$ is a measure of agreement between the predictions of the two tasks on the training data of task t'. In other words, it implies that the classifier trained from task t data correctly classifies $\hat{\rho}_{t,t'}$-fraction of the data points from the task t'. Using the result of Theorem 1, we know that more often than not, $\hat{\rho}_{t,t'}$ is under-estimated. At this point, we assume that the most of the probability mass of the error in $\hat{\rho}_{t,t'}$ estimate is concentrated within $\pm 90°$ of its true value $\rho_{t,t'}$ so that the sign of the estimate is correct with high probability. We can make up for the under-estimation by including the "remaining" unexplained data points from task t' (i.e. the data points that are *novel* w.r.t. the classifier of task t) with some probability (say $q_{t.t'}$) and re-train a new model parameter for task t. In a multi-task setting, we can borrow such data from all other tasks (except task t), thus making up for the under-estimation jointly. As a result of including this borrowed data in the training set of task t, the new model would now correctly classify $\left(\beta_1 \hat{\rho}_{t,t'} + q_{t.t'} \beta_2 \left(1 - \hat{\rho}_{t,t'}\right)\right)$-fraction of data points from the task t' where $\beta_1, \beta_2 \leq 1$. We note that β_1 is related to the capacity of the classifier (such as VC dimension), which is the fraction of previously correctly classified data that are still correctly classified after borrowing data from all other tasks. Similarly, β_2 denote a fraction of the borrowed data from task t' that the new classifier can accurately classify. Let us suppose that by including new data our goal is to bring the estimated relatedness equal to the true relatedness, i.e. $\rho_{t,t'}$, then the following equality is satisfied

$$\rho_{t,t'} = \left(\beta_1 \hat{\rho}_{t,t'} + q_{t.t'} \beta_2 \left(1 - \hat{\rho}_{t,t'}\right)\right), \tag{12}$$

which leads sampling probability used in Eq. (10)

$$q_{t,t'} = \frac{\rho_{t,t'} - \beta_1 \hat{\rho}_{t,t'}}{\beta_2 \left(1 - \hat{\rho}_{t,t'}\right)} = \frac{\eta \hat{\rho}_{t,t'}}{1 - \hat{\rho}_{t,t'}} \tag{13}$$

where $\eta \triangleq \frac{\rho_{t,t'} - \beta_1 \hat{\rho}_{t,t'}}{\beta_2 \hat{\rho}_{t,t'}}$. □

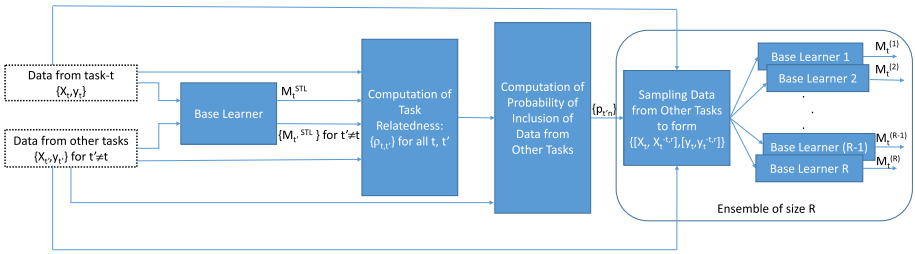

Fig. 3 An illustrative schematic diagram for the proposed ssMTTL framework

If the classifier complexity (e.g., VC dimension) is sufficiently high, the fractions β_1 and β_2 may be close to one and then the parameter η can be interpreted as 'how off is the relatedness estimate from the true relatedness'. Under such assumptions, the value of η varies between 0 to $\frac{(1-\hat{\rho}_{t,t'})}{\hat{\rho}_{t,t'}}$, ensuring that $q_{t,t'}$ lies between 0 and 1. The value of $\eta = 0$ corresponds to "no sharing" between task t and t', the value $\eta = \frac{(1-\hat{\rho}_{t,t'})}{\hat{\rho}_{t,t'}}$ corresponds to "total sharing" and the values between the two extremes correspond to "partial sharing", which often leads to better performance compared to either of the two extremes. Further, we also note that the value of η is inversely proportional to the base classifier complexity, i.e. higher complexity learners for which β_1, and β_2 are closer to 1, require lower η. The appropriate value of η between any two pair of tasks can be learnt independently using a validation set under the framework of model selection. In a scenario where VC-dimension is not large, β_1 and β_2 will be less than one and will cause dependency among $\{\eta, \beta_1, \beta_2\}$ across all the task pairs. In that case learning appropriate values of η for all task-pairs would require joint model selection (Fig. 3).

In practice, it may be convenient to use a single η for all the task-pairs. While this may not guarantee convergence to the true relatedness for all of the task-pairs, model selection via a validation set will ensure that relatedness move closer to their true values for majority of them. A step-by-step procedure for our proposed ssMTTL is provided in Algorithm 1.

Algorithm 1 The proposed ssMTTL algorithm.

1: *Input*: data $\{\mathbf{X}_t, \mathbf{y}_t\}_{t=1}^{T}$, parameter η, parameter σ (for regression), ensemble size (R)
2: *Output*: Ensemble of models $\left\{\mathcal{M}_1^{(r)}, \ldots, \mathcal{M}_T^{(r)}\right\}_{r=1}^{R}$
3: *Initialization:* Initialize $\mathcal{M}_t$ with training of each task independently. Initialize $r = 1$.
4: **while** (r <= R) **do**
5: For task-t, include all its data points. Denote this data as $(\mathbf{X}_t, \mathbf{y}_t)$.
6: For each data point (indexed by n) from all other tasks except task t, compute inclusion probability p_{tn} as in (10).
7: For each data point (indexed by n) from other tasks (except task t), draw a sample u_{tn} from uniform distribution and set $z_{tn} = 1$ if $u_{tn} < p_{tn}$. If $z_{tn} = 1$, include the data point in the training set of task t.
8: Construct an augmented training set for task t as $([\mathbf{X}_t, \mathbf{X}_t^{-t,r}], [\mathbf{y}_t, \mathbf{y}_t^{-t,r}])$ where $(\mathbf{X}_t, \mathbf{y}_t)$ is data from task t and $(\mathbf{X}_t^{-t,r}, \mathbf{y}_t^{-t,r})$ is the data borrowed from other tasks.
9: Using the augmented training set, learn a supervised model (denoted as $\mathcal{M}_t^{(r)}$) for task t as in (2).
10: set $r = r + 1$.
11: **end while**

3.3 Computational complexity

In most cases, the computational complexity of our algorithm is proportional to the complexity of the underlying classification/regression model. In rare cases where the complexity of the underlying model is sub-linear, our algorithm still has linear complexity. For each task t, it is required to compute probability of including data from other tasks. If N_t is the number of data points in task t (where $t = 1 \ldots, T$) and $N = \sum_t N_t$ is the total number of data points across all tasks, then the complexity of computing sample selection probabilities is $\mathcal{O}(N)$. Since each data point is included with a certain probability, which is realized using an ensemble of R models, the overall worst-case complexity becomes $\mathcal{O}(R \times \max(\mathcal{C}(N), N))$ where $\mathcal{C}(N)$ is the complexity of the underlying classification/regression model.

Online MTL requires the model to be updated when new tasks enter the system. For our model, this simply means that the new task can include useful data from the existing tasks while the existing tasks can include useful data from the new task and incrementally update the model. Therefore, as long as a classification or regression model can update its parameters incrementally (i.e. any online classification/regression model), our framework is amenable for online learning.

4 Experiments

We perform experiments with both synthetic data and real data. The experiments with synthetic data enables us to control the data generation process and thus helps studying the behavior of the proposed ssMTTL vis-à-vis other MTL methods. The experiments using real data demonstrate the effectiveness of the ssMTTL for MTL on real-world problems. We compare ssMTTL with a variety of relevant baselines and validate its effectiveness for both classification and regression problems.

4.1 Baseline methods

For both synthetic and real data experiments, we compare ssMTTL with single-task learning and several state-of-art baseline MTL techniques. In addition, we also compare ssMTTL with some transfer learning algorithms. Although both transfer learning and multi-task learning techniques aim to do some form of knowledge transfer, MTL is different from transfer learning in the way that unlike transfer learning, where knowledge transfer happens unidirectionally (from source to target), knowledge transfer in MTL happens bi-directionally for two tasks or multi-directionally for multiple tasks. However, some transfer learning algorithms can be adopted to have such multi-directional knowledge transfer. We adopt three such transfer learning algorithms, namely Co-Training Domain adaptation (CTDA) [11], Adaptive-SVM [59] and Transfer AdaBoost (TrAdaBoost). A brief description of baseline methods is provided below:

- *STL*: Single Task Learning (STL) learns each task independently. We note that the specific model can vary based on which base classifier is used, e.g., STL using logistic regression is referred to as STL(LR). Similarly, STL using SVM, KNN or Random Forest are referred to as STL(SVM), STL(KNN) or STL(RF), respectively.
- *ATL*: Aggregated Task Learning (ATL) pools data from all the tasks together and learns a single-task parameter for all tasks. Similar to STL, the specific model can vary based on which base classifier is used, e.g., ATL using logistic regression, SVM, KNN or Random Forest are referred to as ATL(LR), ATL(SVM), ATL(KNN) or ATL(RF), respectively. This is essentially ssMTTL when no sample selection mechanism is used.

- *MTFL*[2]: Multi-task feature learning (MTFL) learns the relationships between tasks by constraining the task parameters to share a common set of features. They use ℓ_2/ℓ_1 based group-lasso regularization on task parameters. The proposed non-convex formulation is translated to a convex optimization problem with an iterative solution.
- *GMTL* [32]: GMTL partitions the tasks into groups and task parameters of each group are modeled through a subspace. Tasks in a group have a common feature representation. This method learns the group partition and task parameters jointly by optimizing an objective function using an alternating minimization approach. This model facilitates the task parameters to be shared among tasks in a group, but not across the groups.
- *MTRL* [63]: Multi-task relationship learning (MTRL) learns the relationship between tasks using a covariance matrix that allows to capture both positive and negative correlations. It uses a convex formulation for MTL and solution is found using an alternating minimization approach.
- *CTDA* [11]: Co-training for domain adaptation (CTDA) is a technique that can transfer knowledge from a source to a target domain. This algorithm can handle situations where target domains have some additional feature dimensions over the source domain. However, it is developed to be applied in a single source scenario. To adopt it for the multi-task setting, we consider each task as a target task and combine all the other tasks to form a source task.
- *A-SVM* [59]: Adaptive-SVM (A-SVM) is a transfer learning technique that uses SVM as the classifier. It learns the difference of the target task classifier from the source task. When many source tasks are present, a convex combination of source tasks classifier is used. To adopt it to MTL, we learn each task parameter separately by setting it as a target task and taking the average of other tasks as the source task.
- *TrAdaBoost* [14]: Transfer AdaBoost is a transfer learning algorithm that uses a variant of AdaBoost classifier to transfer knowledge from a source domain to a target domain. Similar to MTL algorithms, this algorithm assumes that the source and target data use exactly the same set of features and class labels; however, the distribution of the data in the two domains may be different. To adopt it for the multi-task setting, we consider each task as a target task and combine all the other tasks to form a source task.

Similar to STL and ATL methods, for classification we also have different variants of our proposed ssMTTL as ssMTTL(LR), ssMTTL(SVM), ssMTTL(KNN), ssMTTL(RF) etc depending on the base classifier used. For regression, these variants are ssMTTL(RR) and ssMTTL(SVR) depending on the base regressor used. All the model parameters (η and other parameters relevant to the base learners) are learnt using five-fold cross-validation.

All our experiments were performed in MATLAB 2013a. We implemented our ssMTTL algorithm on top of the implementation of base classifier/regressor models. We used "statistics and machine learning toolbox" in MATLAB for KNN and Naive Bayes, LIBSVM package for SVM (classification) and SVR (regression), and MATLAB Central contribution[2] for Random Forest.

4.2 Experiments with synthetic data

The experiments using synthetic data are performed to illustrate the utility of ssMTTL under *two* different scenarios: (a) when tasks have *varying degree of relatedness*, and (b) when classification tasks need *nonlinear separation boundaries*.

[2] http://au.mathworks.com/matlabcentral/fileexchange/31036-random-forest.

4.2.1 Synthetic data-I : (tasks with varying degree of relatedness)

4.2.2 Data generation

In Synthetic data-I, we generate data for both classification and regression tasks. For both the classification and regression tasks, the generation of $\mathbf{x}$ and task parameters $\mathbf{w}$ follow the same rule, however, $\mathbf{y}$ is generated differently. First we describe the generation process for $\mathbf{x}$ and $\mathbf{w}$, followed by the generation process of y for classification and regressions tasks separately. These data are generated assuming a linear functional form, i.e. either the function is linear as for regression, or the class boundary is linear as for classification. To demonstrate the varying degree of relatedness, we generate two different levels of task relatedness: *perfect* positive/negative task relatedness (an ideal situation for a MTL algorithm), *low-to-high* positive/negative task relatedness's (a more realistic situation). We refer to these datasets as 'Synthetic data-Ia', and 'Synthetic data-Ib', respectively, with suffix (C) denoting classification tasks and (R) denoting regression tasks. For each dataset, we simulate 10 tasks, 30 examples per task and each example lies in a 20-dimensional space.

For 'Synthetic data-Ia', our goal is to create tasks whose relatedness is 100 %, either positive or negative. For this, we create two groups of tasks out of the total 10 tasks. Tasks 1–5 belong to the first group and tasks 6–10 belong to the second group. We randomly generate a task parameter which is shared across all the tasks in the first group and a negative of this task parameter is shared across all the tasks of the second group.

For 'Synthetic data-Ib', our goal is to create tasks whose relatedness is mixed and on average at medium level. For this, similar to 'Synthetic data-Ia', we create two groups of tasks where tasks 1–5 belong to the first group and tasks 6–10 belong the second group. However, this time, for each group (indexed by g), we create 2 random basis vectors from a multi-variate i.i.d. normal distributed vector with mean and standard deviation equal to 0 and 1 respectively, i.e. $\mathbf{b}_1^g, \mathbf{b}_2^g \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We use the two basis vectors of a group to span the task parameters of each task in the group, i.e. $\mathbf{w}_t = [\mathbf{b}_1^g, \mathbf{b}_2^g]\mathbf{a}_t + \delta_t$, ($g = 1$ if $t \in \{1, \ldots, 5\}$ and 2 if $t \in \{6, \ldots, 10\}$) where $\mathbf{a}_t$ is a non-negative coefficient vector such that its entries sums to one, i.e. $||\mathbf{a}_t||_1 = 1$ and $\delta_t \sim \mathcal{N}(\mathbf{0}, 0.1 \times \mathbf{I})$. For both the datasets, the data for t-th task , i.e. $\mathbf{X}_t = \left[\mathbf{x}_1^t, \ldots, \mathbf{x}_{N_t}^t\right]$ are generated as $\mathbf{x}_i^t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Given i-th data vector $\mathbf{x}_i^t$ and the task parameter $\mathbf{w}_t$, y_i^t is generated differently for a regression task and a classification task.

For Synthetic data-Ia(C) and Ib(C), where the suffix (C) refers to classification tasks, the label y_i^t is generated as:

$$y_i^t = \text{sign}\left(\mathbf{w}_t^T \mathbf{x}_i^t + \epsilon_i^t\right),\ \epsilon_i^t \sim \mathcal{N}(0, 0.1) \tag{14}$$

For Synthetic data-Ia(R) and Ib(R), where the suffix (R) refers to regression tasks, the target y_i^t is generated as:

$$y_i^t = \mathbf{w}_t^T \mathbf{x}_i^t + \epsilon_i^t,\ \epsilon_i^t \sim \mathcal{N}(0, 0.1) \tag{15}$$

4.2.3 Experimental results

Using the data generation process described above, we generate 10 different datasets of both types. The left column of Fig. 7 corresponds to Synthetic data-Ia(C) and the right column corresponds to the Synthetic data-Ib(C). Hinton plot is used to represent the task-relatedness matrices, where area of each colored square is proportional to the value of the entry with

Table 1 Average prediction performance of various methods for Synthetic data-Ia(C) and Ib(C) (varying task relatedness)

Dataset	STL	ATL	MTRL	MTFL	GMTL	ssMTTL
Synthetic-Ia(C) (high '+/-'ve relatedness)	0.838 (0.011)	0.489 (0.011)	0.825 (0.012)	0.805 (0.014)	0.876 (0.011)	**0.961** (0.005)
Synthetic-Ib(C) (medium '+/-'ve relatedness)	0.813 (0.014)	0.762 (0.009)	0.799 (0.014)	0.786 (0.015)	0.795 (0.012)	**0.851** (0.012)

The result is reported by averaging the AUC performance over 10 different datasets. The corresponding standard errors are reported in parenthesis. For STL, ATL and ssMTTL, *logistic regression* is used as the base classifier

Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

green indicating positive sign and the red indicating the negative sign. The first row of Fig. 7 shows *true* task relatedness matrix (averaged over 20 different datasets) used for Synthetic data-Ia(C) and Ib(C). The second row of Fig. 7 shows the task relatedness *estimated by STL*. It can be clearly seen that STL-based task relatedness are usually *underestimated*.[3] This is an instance of the theoretical result provided by Theorem 1 in Sect. 3. The third row shows the task relatedness *estimated by ssMTTL*. We can see that task relatedness's estimated by ssMTTL are closer to the true values compared to that estimated by STL. Once again, this is an instance of the theoretical result provided by Theorem 2 in Sect. 3. For the Synthetic-Ia(C) dataset, the mean square error (MSE) between the task relatedness estimates by STL and the true relatedness is 0.19, while the same for ssMTTL is 0.01. Similarly, for the Synthetic-Ib(C) dataset, the mean square error (MSE) between the task relatedness estimated by STL and the true relatedness is 0.14, while the same for ssMTTL is 0.07. Therefore, for both the cases, ssMTTL learns better task relatedness, translating to better prediction performance. Similar trend was also observed for regression tasks involving both Synthetic data-Ia(R) and Ib(R). However, these plots are omitted for brevity.

Table 1 compares the performance of the proposed ssMTTL for classification tasks with several baseline methods using area under the ROC curve (AUC) measure. It can be seen from the table that for Synthetic-Ia(C) dataset, which is ideal for MTL algorithms due to perfect correlations in tasks, the performance of MTL algorithms is clearly better than that of STL. The performance of ssMTTL is more than 8 % better than that of GMTL, which is the closest contender and more than 12 % better than that of STL. ATL suffers greatly due to the strong negative relatedness across the tasks of the two groups. MTFL also suffers due to its strict assumption of a single group of tasks. For Synthetic-Ib(C) dataset, which is a more realistic scenario that MTL techniques often have to deal with, ssMTTL once again improves over STL by around 3.7 % while the other techniques are not able to outperform STL.

Table 2 compares the performance of ssMTTL for regression tasks with other baselines using both the normalized RMSE and Explained variance (R^2). Similar to classification, the Synthetic data-Ia(R) is the most conducive dataset for MTL due to the presence of almost perfect correlation between tasks. All the MTL algorithms perform better than STL for this

[3] The underestimations may be noticed in magnitude of relatedness, irrespective of its sign, i.e. positive relatedness values are often estimated as lower positive values, while negative relatedness values are often estimated as lower negative values.

Table 2 Average prediction performance of various methods for Synthetic data-Ia(R) and Ib(R) (varying task relatedness)

Dataset	STL	ATL	MTRL	MTFL	GMTL	ssMTTL
Synthetic-Ia (high '+/-'ve relatedness)	Explained variance (R^2)					
	0.812 (0.016)	0.041 (0.017)	0.922 (0.011)	0.943 (0.011)	0.967 (0.013)	**0.978** (0.008)
	Normalized RMSE					
	0.368 (0.005)	1.065 (0.009)	0.236 (0.004)	0.196 (0.004)	0.164 (0.003)	**0.124** (0.003)
Synthetic-Ib (medium '+/-'ve relatedness)	Explained variance (R^2)					
	0.802 (0.024)	0.370 (0.031)	0.830 (0.006)	0.781 (0.013)	0.821 (0.006)	**0.890** (0.006)
	Normalized RMSE					
	0.379 (0.001)	0.793 (0.001)	0.362 (0.001)	0.412 (0.001)	0.374 (0.002)	**0.289** (0.001)

The result is reported by averaging both the Normalized RMSE and Explained Variance(R^2) over 10 different datasets. The corresponding standard errors are reported in parenthesis. For STL, ATL and ssMTTL, *ridge regression* is used as the base regressor
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

data; however, ssMTTL performs the best providing almost perfect prediction in terms of both the normalized RMSE and the Explained Variance (R^2). For Synthetic data-Ib(R), the situation turns a bit more challenging due to the presence of moderate correlations between tasks. However, for that dataset also ssMTTL performs the best among all the algorithms compared.

Figure 4a shows the AUC as a function of ensemble size (R) for both the Synthetic data-Ia(C) and Ib(C) averaged over 50 trials. Similarly, Fig. 4b shows both the *normalized RMSE* and *Explained variance* (R^2) for both the Synthetic data-Ia(R) and Ib(R). From all the graphs it can be seen that in most of the cases the performance stabilizes at $R = 20$ and beyond. We consistently observed this point for many datasets and therefore, have chosen to use $R = 40$ for all the experiments in this paper.

To show the strength of our algorithm further, we show the performance of ssMTTL when data are sampled naïvely (all data from other tasks are sampled with a fixed probability) in Table 3. Synthetic dataset-Ia(C) is chosen for this table. Average performance over 50 different splits are presented for both $p_{tn} = 0.1$ and $p_{tn} = 0.9$ for $\forall t \neq t', n$. Since the data is selected with a pre-specified probability, we denote them as ssMTTL($p = 0.1$) and ssMTTL($p = 0.9$), respectively. Also note that in our scheme data from self-task are always selected. An ensemble with $R = 40$ is used for all the cases. From the table we note that this simple scheme of choosing data from other tasks with a single probability value does not perform better than both STL and ATL. As expected, ssMTTL($p = 0.1$) performs close to STL and ssMTTL($p = 0.9$) performs close to ATL and both of them are between the range of STL and ATL. On the contrary, ssMTTL with our proposed scheme of computing probability (Eq. (10)) performs much better than both the STL and ATL. This clearly demonstrate that the benefit of our algorithms comes only from the way the probabilities are computed and not from the ensemble. Ensemble is only a way to include a data with a non-perfect and non-zero probability in a classifier. A similar trend was also observed for the regression datasets; however, the results are omitted for brevity.

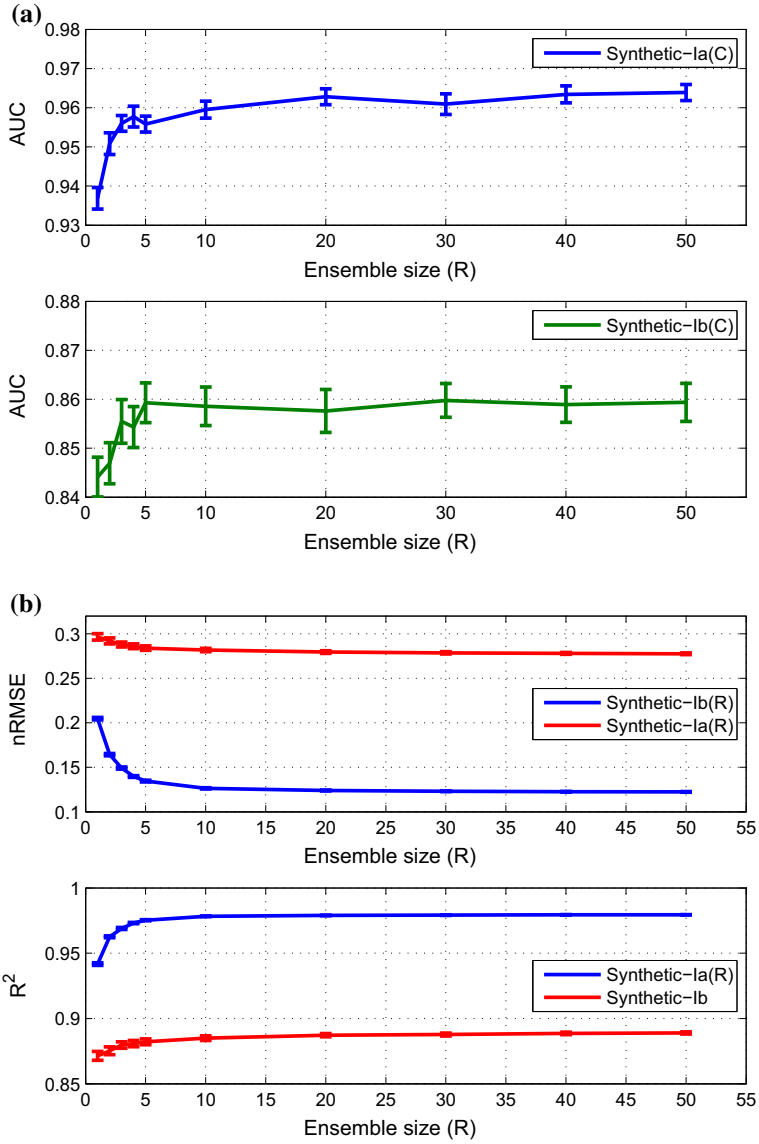

Fig. 4 Performance versus ensemble size (R) for both Synthetic data-Ia and Ib (both R and C) averaged over 50 random splits of training and test data. Standard errors are reported as *error bars*. **a** Synthetic data-Ia(C) and Ib(C). **b** Synthetic data-Ia(R) and Ib(R)

Figures 5 and 6 show the prediction performance as a function of η for both classification and regression task for Synthetic data-I. As Synthetic data-Ia is the most conducive dataset for MTL, we observe that for both the classification (Fig. 5) and regression tasks (Fig. 6), the performance almost saturates at higher η. The performance for regression falls slightly, however, no such fall was observed for the classification tasks even at a very high η. Both the saturation or only a slight fall is expected as the tasks have near-perfect correlation, and over-correction can only bring to the perfect correlation, which may not be detrimental to the

Table 3 The effect of probability p_{tn} on the performance on Synthetic data-Ia(C)

Method	ATL	ssMTTL ($p = 0.9$)	ssMTTL ($p = 0.1$)	STL	ssMTTL
AUC (std err)	0.489 (0.011)	0.552 (0.011)	0.835 (0.010)	0.838 (0.011)	**0.961** (0.005)

ssMTTL($p = 0.1$) implies that the probability of inclusion of any data from all other tasks is set equally to 0.1, and for ssMTTL($p = 0.9$) that is set at 0.9. ssMTTL in the last column uses the probabilities computed by Eq 10 and are different for different data. In all the cases self-task data are always selected
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

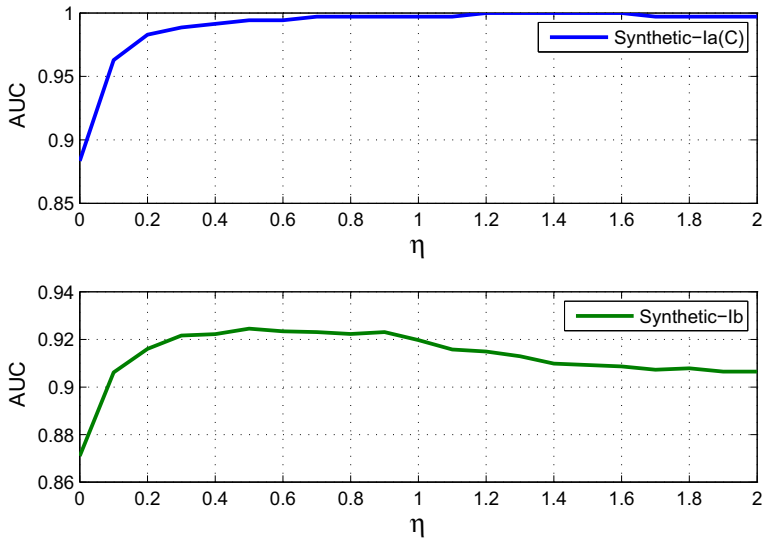

Fig. 5 AUC versus η for both Synthetic data-Ia(C) and Ib(C). This is shown for ssMTTL with logistic regression as the base classifier

performance in this scenario. For Synthetic-Ib, we can clearly see that performance peaks at a certain value of η and falls thereafter. This is observed irrespective of tasks type and the performance measure chosen. This is due to the fact that there exist an optimal value of η where the correction for task-to-task correlation happens at the right level. An η smaller than the optimal value would under-correct and an η more than that would over-correct the task-to-task correlations. In both cases, performance would fall short of what can be achieved when corrections were made just right.

4.2.4 Synthetic data-II: (tasks with nonlinear separation boundaries)

4.2.5 Data generation

For Synthetic data-II, we again generate 10 classification and regressions tasks. For each task, we simulate data where either the class boundaries are *nonlinear* or the regressors are *nonlinear* functions. Once again, we generate two groups of tasks so that the tasks 1-5 are part of group-1 and the tasks 6-10 are part of group-2. This is done to create both related and unrelated tasks. Tasks within a group are strongly related meaning that the task parameters are almost same up to a small noise, i.e. for $t \in \{1, \ldots, 5\}$, $\mathbf{w}_t = \mathbf{b}^1 + \delta_t$ and for $t \in \{6, \ldots, 10\}$,

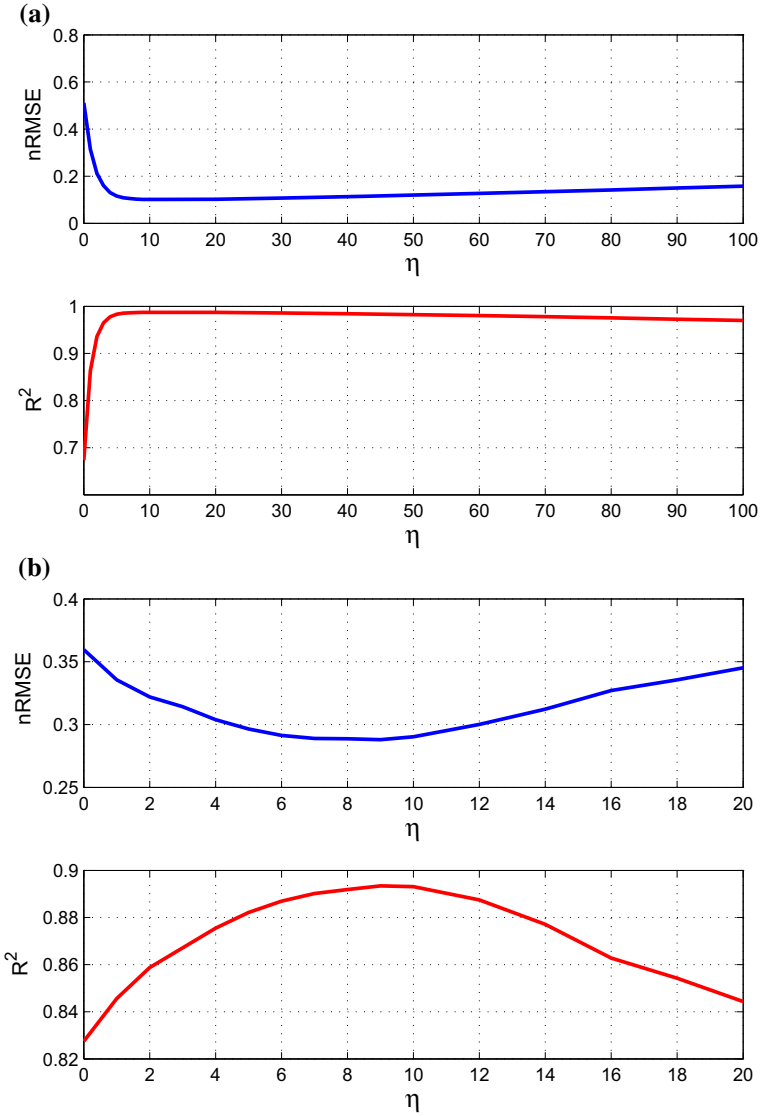

Fig. 6 Normalized RMSE and Explained variance (R^2) versus η for both Synthetic data-Ia(R) and Ib(R). These plots are shown for ssMTTL with ridge regression as the base regressor. **a** Synthetic data -Ia(R). **b** Synthetic data -Ib(R)

$\mathbf{w}_t = \mathbf{b}^2 + \delta_t$, where $\mathbf{b}^1, \mathbf{b}^2$ are the basis vectors for group-1 and group-2, respectively, and sampled as $\mathbf{b}^1, \mathbf{b}^2 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The noise vector δ_t is sampled from a multi-variate Gaussian as $\delta_t \sim \mathcal{N}(\mathbf{0}, 0.1 \times \mathbf{I})$. All the task parameters are generated in a 4-dimensional space. For each task, i-th example is simulated by drawing a vector $\mathbf{x}_i^t = \left[\mathbf{x}_{i1}^t, \mathbf{x}_{i2}^t\right]^T$ from a *bi-variate* normal distribution. However, to simulate a nonlinear boundary in 2-dimensional space, the label y_i^t for Synthetic data-II(C) is generated as the following,

$$y_i^t = \text{sign}\left(\mathbf{w}_t^T \left[\mathbf{x}_{i1}^t, \mathbf{x}_{i2}^t, \left(\mathbf{x}_{i1}^t\right)^2, \left(\mathbf{x}_{i2}^t\right)^2\right]^T + \epsilon_i^t\right), \ \epsilon_i^t \sim \mathcal{N}(0, 0.1). \tag{16}$$

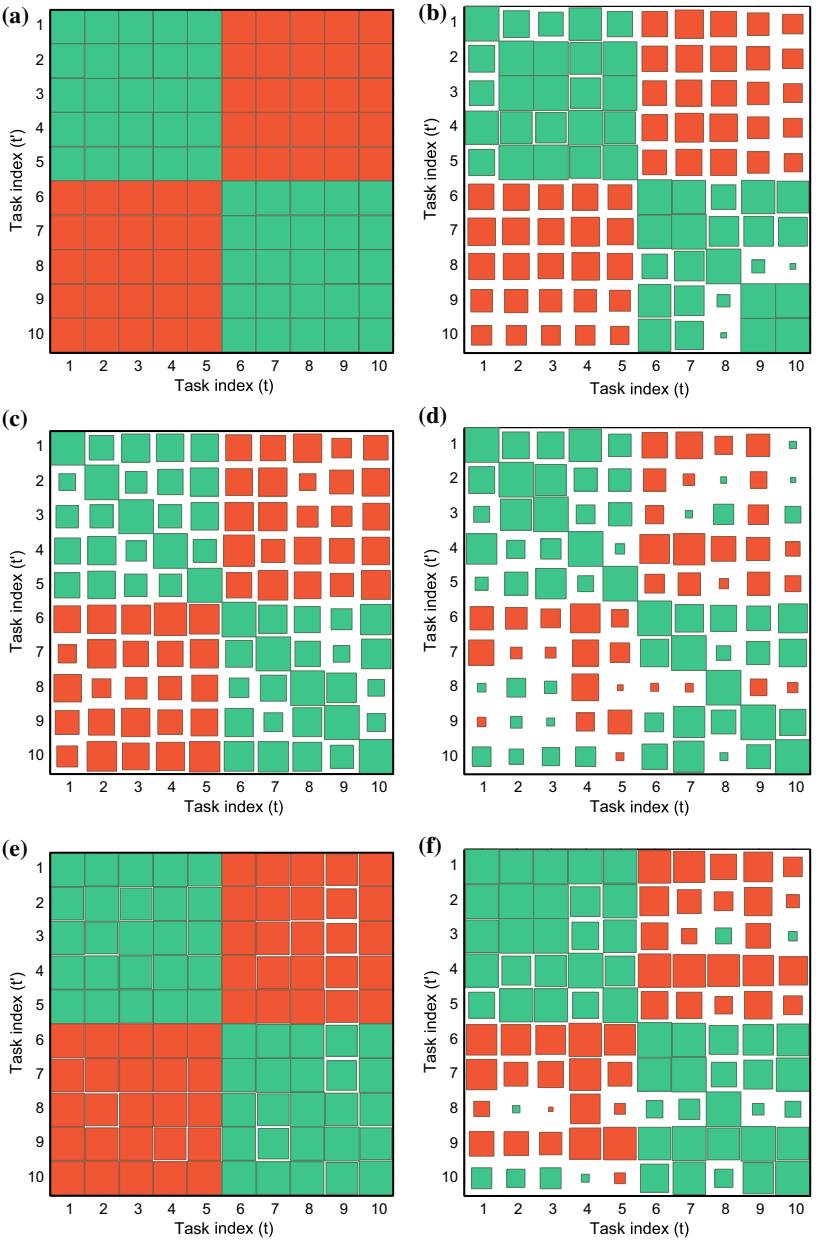

Fig. 7 Estimation of task relatedness by STL and ssMTTL for synthetic data-I (varying task relatedness) classification. The *left* column corresponds to Synthetic data-Ia(C) and the *right* column corresponds to the Synthetic data-Ib(C). The *first* row shows *true* task relatedness, the *second* row shows the relatedness estimated by *STL* and the *third* row shows the relatedness estimated by *ssMTTL*. The task-relatedness matrices are shown using Hinton plots where the area of the colored squares are proportional to the entries (maximum size of a square indicates a value of 1). The color green indicates a positive value and red indicates a negative value. **a** Synthetic data-Ia(C): true task relatedness. **b** Synthetic data Ib(C): true task relatedness. **c** Synthetic data-Ia(C): task relatedness after STL. **d** Synthetic data-Ib(C): task relatedness after STL. **e** Synthetic data-Ia(C): task relatedness after ssMTTL. **f** Synthetic data-Ib(C): task relatedness after ssMTTL

Similarly, to simulate a nonlinear functional form for the regressor for Synthetic data-II(R) the target y_i^t is generated as

$$y_i^t = \mathbf{w}_t^T \left[\mathbf{x}_{i1}^t, \mathbf{x}_{i2}^t, \left(\mathbf{x}_{i1}^t\right)^2, \left(\mathbf{x}_{i2}^t\right)^2\right]^T + \epsilon_i^t,\ \epsilon_i^t \sim \mathcal{N}(0, 0.1). \tag{17}$$

Although the data generated via the above approach is linear functional in the 4-dimensional feature space formed by $\left(\mathbf{x}_{i1}^t, \mathbf{x}_{i2}^t, \left(\mathbf{x}_{i1}^t\right)^2, \left(\mathbf{x}_{i2}^t\right)^2\right)$, the classification boundary (or the function) becomes nonlinear in 2-dimensional feature space formed by $\left(\mathbf{x}_{i1}^t, \mathbf{x}_{i2}^t\right)$. All the methods use data $\{(\mathbf{X}_t, \mathbf{y}_t)\}_{t=1}^{10}$ where $\mathbf{X}_t = \left[\mathbf{x}_1^t, \ldots, \mathbf{x}_{N_t}^t\right]$ and $y_t = \left[y_1^t, \ldots, y_{N_t}^t\right]$. Number of examples in each task is 30.

4.2.6 Experimental results

We use STL, ssMTTL to estimate task parameters for the above generated dataset. Figure 8 shows the *true* task relatedness, task relatedness estimated by STL(LR), ssMTTL(LR), STL(SVM-poly2) and ssMTTL(SVM-poly2) in the order of sub-figures (a) to (e). We can see that STL(LR) using a linear classifier could estimate the task relatedness only partially because of the underestimation. Since the data are not linearly separable, even ssMTTL(LR) could not compensate the underestimations in relatedness. This suggests the need of using a nonlinear classifier. When we use STL(SVM-poly2) model, which is STL using a SVM classifier with polynomial kernel of degree 2, we can see that estimations in task relatedness have improved and they improved further when we use ssMTTL(SVM-poly2) model, i.e. the multi-task learning model using a SVM-poly2 as base classifier. Quantitatively, the mean square error between the true task relatedness and the one estimated by STL(SVM-poly2) is 0.07, which is reduced to 0.03 by using ssMTTL(SVM-poly2).

The underestimations of task relatedness directly affect the prediction performances as seen in Table 4. We compare the performance of ssMTTL with STL and ATL using three base classifiers– LR, SVM with polynomial kernel of degree 2 (referred to as SVM-poly2) and SVM with RBF kernel (referred to as SVM-RBF). Clearly, both the SVM-based ssMTTL classifiers improve upon their STL counterparts. However, this is not the case for LR classifier as its performance for both STL and ssMTTL is similar and not up to those of SVM classifiers. This suggests that when data are nonlinearly separable, using SVM classifier can achieve better performance and ssMTTL(SVM) can lift the performance even further. For KNN and Random Forest, where it is possible to learn more flexible nonlinear boundaries, the performance by both STL and ssMTTL are stellar. However, for both the base classifiers, ssMTTL still performs the best. Finally, the performance of ATL for all five base classifiers (see Table 4) show that simple pooling of data from all the tasks degrades the performance when there are unrelated tasks in the pool.

Table 5 presents the comparative performance of ssMTTL with respect to both the STL and ATL for different base regressors on Synthetic data-II(R). Since the data are nonlinear by design, ssMTTL with nonlinear base regressors such as SVR with Polynomial of degree 2 (poly2) kernel and RBF kernel performed significantly better than the linear ridge regression in terms of both *Explained variance* (R^2) and *normalized RMSE*. Further, since the data have been originally generated using a polynomial of degree 2 function, the ssMTTL with SVR-poly2 as the base regressor performs the best and ssMTTL improves over both the STL and ATL with a satisfactory margin. Finally, it shows that ATL which combines data without any consideration for task relatedness fails totally.

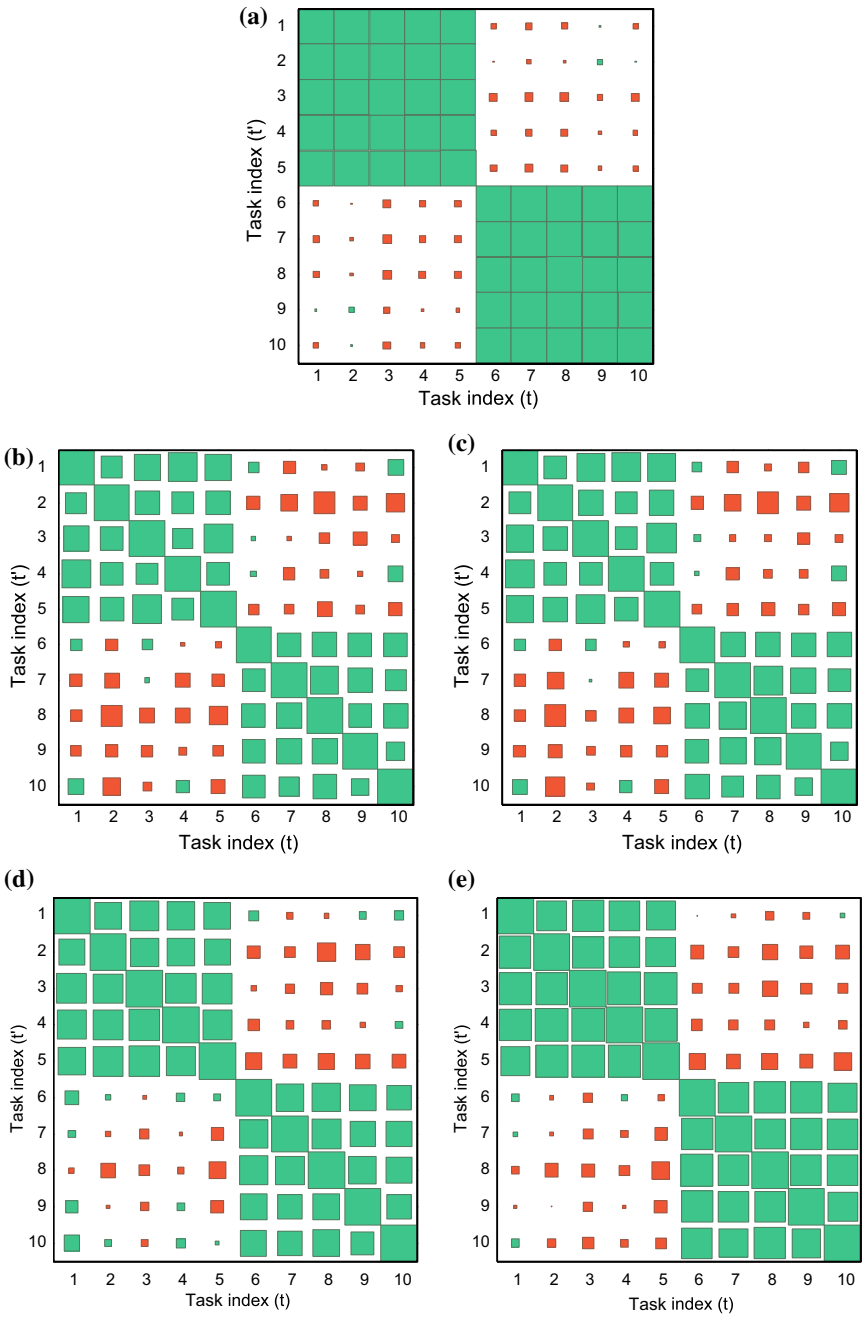

Fig. 8 Estimation of task relatedness by STL and ssMTTL using logistic regression and SVM as base classifiers for synthetic data-II (nonlinearly separable case) classification: task relatedness shown are **a** true , and using, **b** STL(LR), **c** ssMTTL(LR), **d** STL(SVM-poly2), and **e** ssMTTL(SVM-poly2). The task-relatedness matrices are shown using Hinton plots where the area of the *colored squares* are proportional to the entries (maximum size of a *square* indicates a value of 1) with the *color green* indicating a positive value and *red* indicating a negative value

Table 4 Average prediction performance of various methods for Synthetic data-II(C) (tasks have nonlinearly separable data)

Base Classifier	STL	ATL	ssMTTL
LR	0.700 (0.016)	0.645 (0.018)	**0.701** (0.017)
SVM-poly2	0.828 (0.013)	0.731 (0.005)	**0.866** (0.013)
SVM-RBF	0.805 (0.016)	0.775 (0.016)	**0.831** (0.011)
KNN	0.909 (0.003)	0.788 (0.005)	**0.951** (0.002)
Random Forest	0.919 (0.002)	0.729 (0.006)	**0.936** (0.002)

The result is reported by averaging the AUC performance over 10 different datasets. The corresponding standard errors are reported in parenthesis
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

Table 5 Average prediction performance of various methods for Synthetic data-II(R) (tasks with nonlinear regressors)

Base classifier	STL	ATL	ssMTTL
	Explained variance ($R^{2)}$)		
RR	0.340 (0.016)	0.141 (0.009)	**0.342** (0.016)
SVR-poly2	0.820 (0.011)	0.272 (0.017)	**0.830** (0.011)
SVR-RBF	0.682 (0.016)	0.252 (0.015)	**0.696** (0.015)
	Normalized RMSE		
RR	0.666 (0.018)	0.840 (0.018)	**0.663** (0.018)
SVR-poly2	0.238 (0.006)	0.780 (0.019)	**0.228** (0.005)
SVR-RBF	0.384 (0.013)	0.775 (0.019)	**0.373** (0.012)

The result is reported by averaging the Explained variance (R^2) over 10 different datasets. The corresponding standard errors are reported in parenthesis
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

4.3 Experiments with real data

4.3.1 Datasets

We evaluate the proposed ssMTTL on the following *classification* and *regression* datasets.

Classification Datasets

– **Cancer Data:** The Cancer dataset is obtained from a large regional hospital in Australia.[4] There are eleven different cancer types in this data recorded from patients visiting the hospital during 2010–2012. Patient data come from a special cancer registry (ECO). These data contain a variety of information such as patient demographics, medical conditions in patient's previous visits recorded using ICD-10 codes[5] and various tumor-specific information (e.g., size, number of sites etc.). The overall feature dimension of this dataset

[4] Ethics approval obtained through University and the hospital—12/83.

[5] http://www.who.int/classifications/icd10/.

is 22. The number of patients combined across different cancer types is 668 and the number of patients per task ranges from 14 to 120. Each task involves 1-year survival prediction (a binary classification problem) where we have to predict whether a patient survives at least unto 1 year after a positive cancer diagnosis or not. Different cancers have different survival rate with breast cancer being the least lethal with a high 96.7 % survival rate and cancer in upper gastro is the most lethal with only 41 % of the patients surviving into the second year after detection. Many of these cancer types share several features and their effect to survival is correlated. To exploit their statistical sharing strength, we jointly model these tasks as a binary multi-task classification problem.

– **Landmine Data:** This is a benchmark dataset that has been previously used in [58]. It contains examples collected from various landmine fields. Each example is represented in a 9-dimensional feature space. The task is to classify each example in one of the two classes : 'landmine' and 'clutter', denoted by 1 and 0 respectively. The feature vectors are concatenation of various aggregated features such as moment features, correlation-based features, energy ratio feature and spatial variance feature, derived from radar images. Number of examples per task ranged from 445 to 690 and on an average 6 % of the examples are positive. Following [30,58], we jointly model 19 tasks of the dataset.

Regression Datasets

– *School Data:* This dataset comes from the Inner London Education Authority and has been previously used in many previous works on multi-task learning such as [2,27,33]. It consists of examination scores of 15362 students from 139 secondary schools in London during 1985-1987. The data for each student are represented as a 26-dimensional feature vector containing year of examination, school and student-specific attributes. Number of students in each school ranged from a low of 22 to a high of 251. Average standard deviation of scores over all the schools is 11.85. We treat the prediction of examination score for each school data as a task and the data from each student as an example.
– *Computer Survey Data:* This dataset was introduced in [38] and later used by [2] for MTL. This dataset contains ratings of 20 personal computers by 190 students. Each computer was rated by students based on 13 binary features (amount of memory (RAM), hard disk capacity , screen size, CPU speed, cache size, CD-ROM, telephone hotline, price, availability, color, warranty, guarantee and software) and assigned a score on a scale of 0-10 indicating their likelihood of buying this product. We treat 13-dimensional rating vectors corresponding to each student as a task, giving rise to 190 tasks. The average of all the ratings in the dataset is 4.75 with average of standard deviation being 1.49. Since we expect these tasks to be related, we jointly model them under the setting of multi-task learning.

4.3.2 Experiments with cancer data

We split the cancer data into 70 % as training and 30 % as testing. Since the classes are highly imbalanced for this data, the splitting is performed per class basis. The Table 6 shows the average AUC for STL, ATL and MTL for 50 different splits, when ssMTTL is applied over different base classifiers. Irrespective of the base classifier used, ssMTTL always performed better than its STL counterpart. Except for the SVM, ssMTTL performs the best for all other classifiers when compared between STL, ATL and ssMTTL. Even with a simple base classifier such as KNN ($K = 3\,\%$ with cosine similarity), we are able to gain considerably by using ssMTTL (AUC 0.677) over the STL (AUC 0.653). Using ssMTTL over Logistic Regression (AUC 0.715) also improves the performance over the STL (AUC 0.698). For both these

Table 6 AUC of prediction on cancer dataset by ssMTTL when different base classifiers are used

Base classifiers	STL	ATL	ssMTTL
LR	0.698 (0.005)	0.723 (0.005)	**0.744** (0.005)
KNN	0.653 (0.005)	0.669 (0.005)	**0.677** (0.006)
SVM-RBF	0.671 (0.007)	**0.771** (0.005)	0.769 (0.005)
Random Forest	0.780 (0.005)	0.643 (0.007)	**0.799** (0.005)

70 % of the data are used for training and the rest for testing. The reported AUC is the average performance over 50 random splits. The corresponding standard errors are shown in parenthesis
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

classifiers, ATL, which is pooling data from all the tasks resulted in performance lower than ssMTTL. Using ssMTTL over SVM (AUC 0.722) improves performance over STL (AUC 0.671), but surprisingly, SVM with ATL (AUC 0.771) works better than ssMTTL. However, out of all four classifiers, using ssMTTL over Random Forest provides the best performance (AUC 0.799). Interestingly, in contrast to SVM, Random Forest suffers greatly when ATL (AUC 0.643) is used. Table 8 shows the average of the absolute relatedness between the tasks after both STL and ssMTTL for different base classifiers. Evidently, for all the base classifiers the average absolute relatedness are higher for ssMTTL than STL, providing an empirical evidence to the Theorem 2. Table 9 shows the comparative performance of the best ssMTTL classifier (from Table 6) with three other MTL and two transfer learning baselines. The ssMTTL(Random Forest) is able to outperform all the five baselines by a good margin. Interestingly, both the transfer learning methods achieved below par performance.

Table 7 presents detailed task-wise breakup of the prediction performance. STL and ssMTTL performance is provided in the consecutive columns for all four base classifiers - Logistic regression, KNN, SVM with RBF kernel and Random Forest. For all of them ssMTTl provided better performance on most of the tasks. For both Random Forest and SVM, ssMTTL provided better or same performance for 10 out of 11 tasks. This demonstrates that the gain in overall task-averaged performance comes from the improvement made in the prediction of most of the tasks.

The Fig. 9 shows the AUC of ssMTTL(LR) with respect to the model parameter η where the value of η is varied from 0 to 2 at a step of 0.1, with $\eta = 0$ implying no sharing between tasks *i.e.* STL. The experiment is performed over only a single split to show the indicative behavior of performance with respect to η. It shows that the performance peaks at η =0.1, thereafter, slowly decreases. Similarly for KNN and Random Forest, the performance peak at different η and falls thereafter. For SVM, since ATL performs the best, we see that the AUC improves almost monotonically with η. Further, we show the average performance of ssMTTL with different base classifiers as a function of the ensemble size (R) over 50 random splits in Fig. 10. As expected the performance stabilizes at $R = 20$ and beyond. This further justifies our choice of $R = 40$.

4.3.3 Experiments with landmine data

We split the landmine data into 30 % as training and 70 % for testing. This way of splitting is done to show the effectiveness of MTL over STL in the regimen of small training data. Similar to the Cancer data, to handle the existing imbalance in the classes, the splitting is performed per class basis. The Table 10 shows the average AUC for STL, ATL and ssMTTL

Table 7 Task-wise AUC of prediction on Cancer dataset by ssMTTL when different base classifiers are used

	STL (LR)	ssMTTL (LR)	STL (KNN)	ssMTTL (KNN)	STL (SVM)	ssMTTL (SVM)	STL (RF)	ssMTTL (RF)
Task-1	**0.956**	0.721	0.625	**0.644**	0.591	**0.719**	0.814	**0.833**
Task-2	0.368	**0.818**	0.699	**0.706**	0.776	**0.812**	0.777	**0.810**
Task-3	0.709	**0.919**	**0.688**	0.668	0.605	**0.818**	0.844	**0.887**
Task-4	**0.752**	0.672	0.661	**0.674**	0.652	**0.682**	0.699	**0.706**
Task-5	**0.886**	0.787	**0.559**	0.552	**0.777**	0.748	0.788	**0.808**
Task-6	0.646	**0.699**	0.622	**0.636**	0.650	**0.689**	0.684	**0.708**
Task-7	0.512	**0.761**	0.738	**0.744**	0.726	**0.821**	**0.837**	0.830
Task-8	0.681	**0.875**	0.765	**0.843**	0.872	**0.907**	0.885	**0.892**
Task-9	**0.733**	0.427	**0.500**	**0.500**	0.500	**0.508**	0.540	**0.574**
Task-10	0.794	**0.910**	0.786	**0.882**	0.774	**0.980**	0.950	**0.982**
Task-11	0.571	**0.593**	0.542	**0.602**	0.453	**0.797**	**0.763**	**0.763**
Better or same	4	**7**	3	**9**	1	**10**	2	**10**

70 % of the data is used for training and the rest for testing. The reported AUC is the average performance over 50 random splits. Comparison is made between the STL and ssMTTL version of the same base classifier. Bold indicates better or same performance between them. The last row summarizes the number of times STL and ssMTTL was better or same

Table 8 Average of absolute task relatedness of STL and ssMTTL for different base classifiers from experiments with the Cancer dataset.

Base classifiers	STL	ssMTTL
LR	0.402 (0.005)	**0.435** (0.005)
KNN	0.260 (0.005)	**0.327** (0.006)
SVM-RBF	0.384 (0.007)	**0.462** (0.005)
Random Forest	0.386 (0.005)	**0.436** (0.006)

For all the classifiers, ssMTTL improves the relatedness over STL. The results are averaged over 50 random splits. The corresponding standard errors are reported in parenthesis
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

Table 9 AUC for prediction on Cancer dataset by ssMTTL, three MTL baselines (MTFL, GMTL and MTRL) and three transfer learning baselines (CTDA, A-SVM and TrAdaBoost)

ssMTTL (LR)	ssMTTL (RF)	MTL baselines			Transfer Learning baselines		
		MTFL	GMTL	MTRL	CTDA	A-SVM	Tr-AdaBoost
0.744 (0.005)	**0.799** (0.005)	0.727 (0.006)	0.723 (0.007)	0.739 (0.005)	0.633 (0.008)	0.650 (0.006)	0.694 (0.006)

The reported AUC is the average performance over 50 random splits. The corresponding standard errors are shown in parenthesis
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

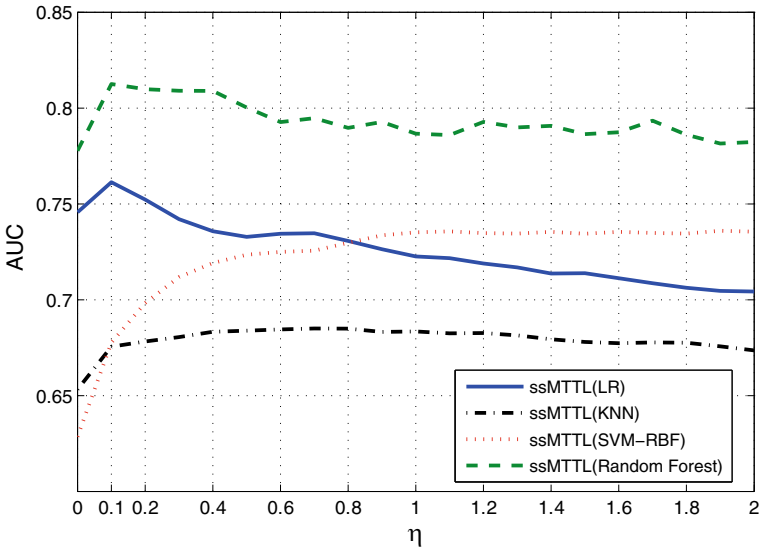

Fig. 9 AUC of ssMTTL on the Cancer dataset as a function of the parameter η for various base classifiers

for 50 different splits for different base classifiers. Irrespective of the base classifier used, the ssMTTL is able to perform the best. For all the classifiers, ssMTTL version is better than both the STL and ATL version of the algorithm. Out of all four classifiers used, the ssMTTL

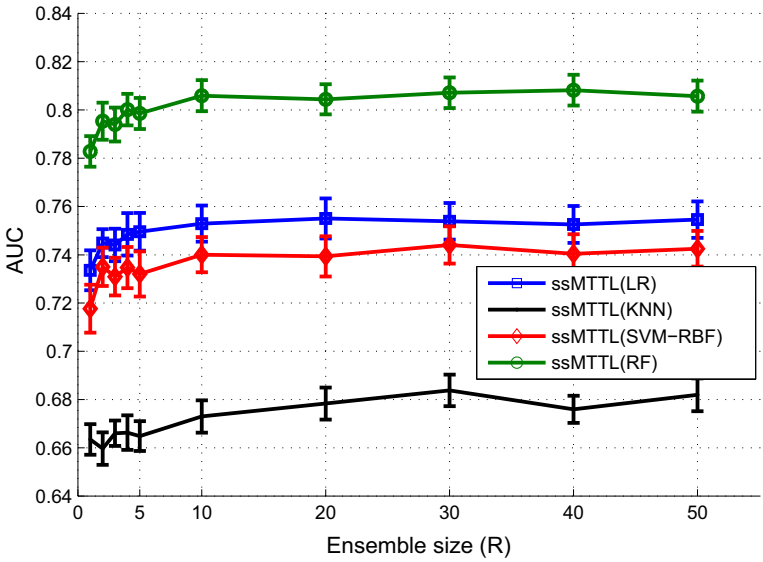

Fig. 10 AUC versus Ensemble size (R) for Cancer dataset averaged over 50 random training-test splits. Standard errors are reported as error bars

Table 10 AUC of prediction on Landmine dataset by ssMTTL when different base classifiers are used

Base classifiers	STL	ATL	ssMTTL
LR	0.654 (0.005)	0.747 (0.005)	**0.765** (0.005)
KNN	0.657 (0.007)	0.649 (0.006)	**0.702** (0.006)
SVM-RBF	0.740 (0.005)	0.752 (0.005)	**0.765** (0.005)
Random Forest	0.752 (0.005)	0.735 (0.006)	**0.767** (0.006)

Only 30 % of the data is used for training and the rest for testing. The reported AUC is the average performance over 50 random splits. The corresponding standard errors are shown in parenthesis
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

version of Random Forest performed the best with the SVM and LR version being joint second. Table 13 shows the comparison of the best ssMTTL classifier (from Table 6) with other MTL and transfer learning baselines. The ssMTTL(Random Forest) version is able to outperform all three MTL and two transfer learning baselines by a significant margin. Table 11 shows the task-wise breakup of comparative performance between the STL and ssMTTL under different base classifiers. The performance is deemed similar if $\Delta\text{AUC} \leq 0.01$. We can clearly see that under all the base classifiers, ssMTTL was able to maintain a similar or provide better performance for 65-75 % of the tasks.

Table 12 shows the average absolute task relatedness for both STL and ssMTTL using different base classifiers. For all the cases the relatedness increased when ssMTTL is used over STL. This results in improved estimation of task parameters, as reflected via improved performance in Table 10.

We also study the performance of the ssMTTL(RF) with respects to the model parameter η. The Fig. 11 shows the AUC by ssMTTL with respect to η. The value of η is varied from

Table 11 Comparative performance of STL and ssMTTL for different base classifiers on Landmine dataset

	LR	KNN	SVM	RF
Similar (within ± 0.01)	2	4	3	8
ssMTTL $<$ STL $- 0.01$	4	4	6	4
ssMTTL $>$ STL $+ 0.01$	13	11	10	7

Performance is similar when the AUC difference is within ± 0.01, otherwise better if difference > 0.01 and worse if < 0.01
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

Table 12 Average of absolute task relatedness of STL and ssMTTL for different base classifiers on Landmine dataset

Base classifiers	STL	ssMTTL
LR	0.472 (0.005)	**0.512** (0.005)
KNN	0.360 (0.005)	**0.415** (0.006)
SVM-RBF	0.424 (0.007)	**0.461** (0.005)
Random Forest	0.478 (0.005)	**0.572** (0.006)

For all the classifiers ssMTTL improves the relatedness over STL. The results are averaged over 50 random splits. The corresponding standard errors are shown in parenthesis
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

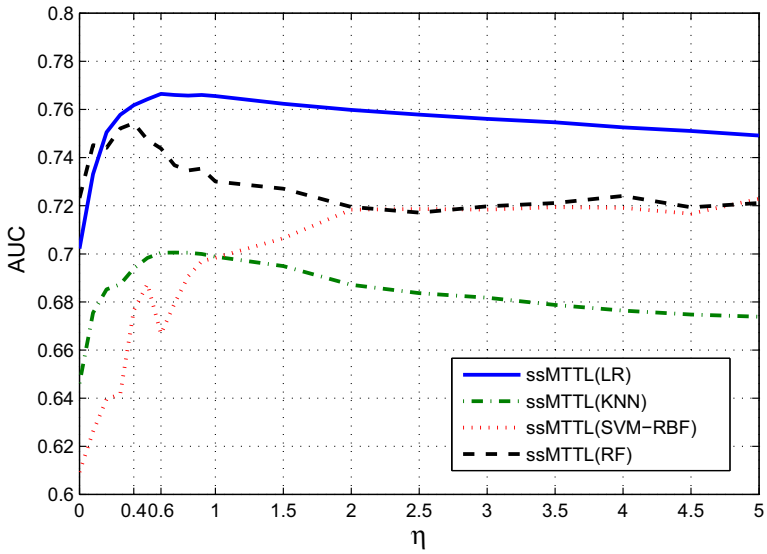

Fig. 11 AUC of ssMTTL on the landmine dataset as a function of the parameter η for various base classifiers

0 to 5, with $\eta = 0$ implying STL. The experiment is performed over only a single split to show the indicative behavior of performance with respect to η. For all the base classifiers, the performance peaks at a certain value of η, when correction of the task correlation is optimal,

Table 13 AUC for prediction on Landmine dataset by ssMTTL, three MTL baselines (MTFL, GMTL and MTRL) and three transfer learning baselines (CTDA, A-SVM and TrAdaBoost)

ssMTTL (LR)	ssMTTL (RF)	MTL baselines			Transfer Learning baselines		
		MTFL	GMTL	MTRL	CTDA	A-SVM	Tr-AdaBoost
0.765 (0.005)	**0.767** (0.006)	0.708 (0.006)	0.702 (0.007)	0.727 (0.006)	0.736 (0.005)	0.608 (0.007)	0.655 (0.008)

The reported AUC is the average performance over 50 random splits. The corresponding standard errors are shown in parenthesis
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

Table 14 Experimental results for regression datasets: The result is reported in terms of Explained variance (R^2) averaged over all tasks and 50 random trials

Dataset	STL(RR)	ATL(RR)	MTFL(RR)	GMTL(RR)	MTRL(RR)	ssMTTL(RR)
(a)						
School	Explained variance (R^2)					
	0.232 (0.000)	0.230 (0.000)	0.254 (0.000)	0.252 (0.009)	0.250 (0.000)	**0.258** (0.009)
	Normalized RMSE					
	0.778 (0.000)	0.794 (0.000)	0.778 (0.000)	0.765 (0.000)	0.776 (0.000)	**0.763** (0.000)
Computer survey	Explained variance (R^2)					
	0.261 (0.000)	0.229 (0.000)	0.302 (0.000)	0.301 (0.009)	0.274 (0.000)	**0.329** (0.009)
	Normalized RMSE					
	0.743 (0.000)	0.788 (0.000)	0.685 (0.000)	0.691 (0.009)	0.725 (0.000)	**0.666** (0.009)
Dataset	STL (SVR)	ATL (SVR)	ssMTTL (SVR)			
(b)						
School	Explained variance (R^2)					
	0.211 (0.000)	0.233 (0.000)	**0.256** (0.000)			
	Normalized RMSE					
	0.794 (0.000)	0.791 (0.000)	**0.770** (0.000)			
Computer survey	Explained variance (R^2)					
	0.221 (0.000)	0.205 (0.000)	**0.269** (0.01)			
	Normalized RMSE					
	0.754 (0.000)	0.808 (0.000)	**0.660** (0.009)			

The numbers in parenthesis are the standard errors. In (a), we compare the results of ssMTTL with STL, ATL and the MTL baselines trained using *ridge regression (RR)* as the base model. In (b), we report the results of nonlinear models: STL, ATL and ssMTTL trained using *support vector regression* (RBF kernel) as the base model
Bold indicates that the performance indicated by the number is the best compared to the corresponding performances shown for the other methods

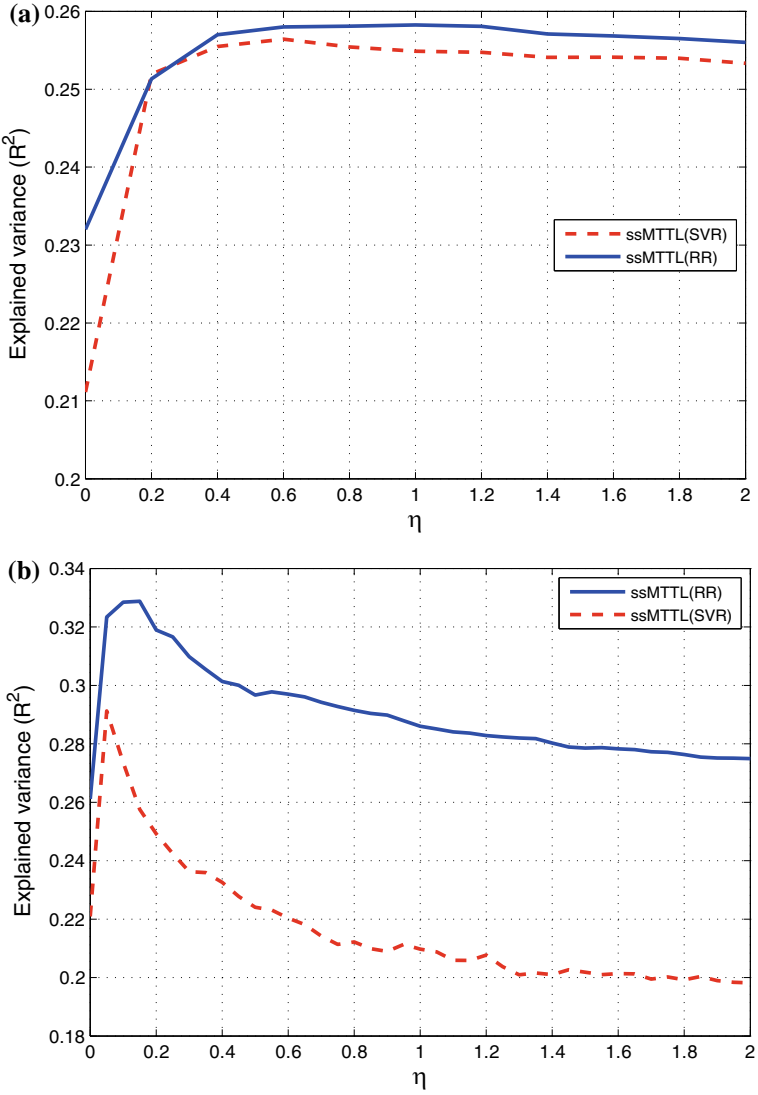

Fig. 12 Explained variance (R^2) for both ssMTTL (Ridge regression) and ssMTTL(SVR) as a function of the parameter η for **a** School dataset, and **b** Computer survey dataset

and then falls thereafter due to over-correction. Different algorithms peak at different values of η. Interestingly, ssMTTL(SVM-RBF) peaks at $\eta = 2$ and falls only slightly thereafter.

4.3.4 Experiments with regression datasets

For school dataset, following [2] each categorical feature is replaced with a set of binary features resulting in a total of 25 features. A term is further added to account for bias terms. We then divide students from each school into 70 % for training and 30 % for testing. Number of students in each school are different: on average there are about 70 students per school

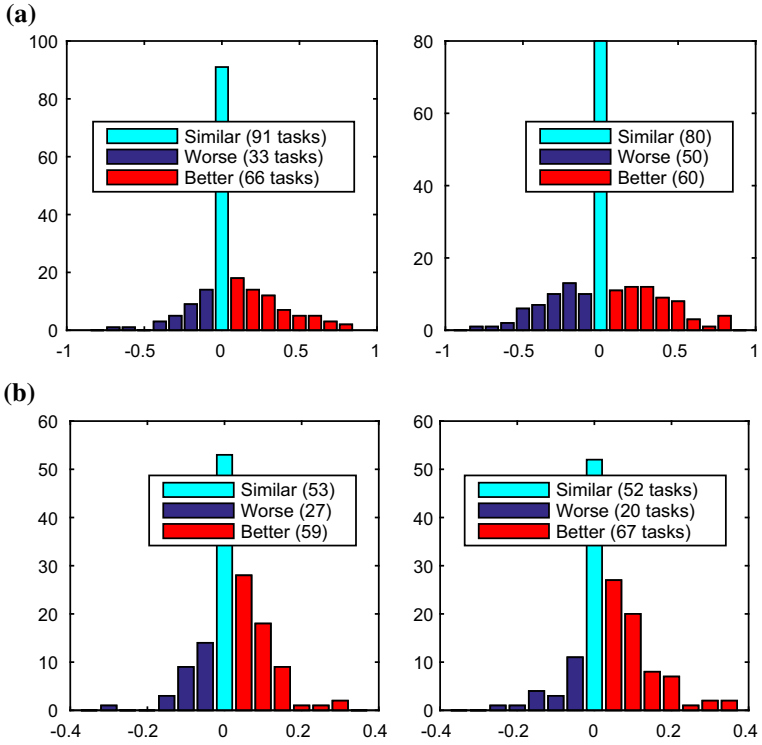

Fig. 13 Task-wise comparative results between STL and ssMTTL for Computer and School datasets, respectively. Tasks are divided into three groups where STL and ssMTTL performance is termed similar if difference of their R^2 performance, $\Delta R^2 \leq \pm 0.05$, better if R^2 by ssMTTL is at least 0.05 greater than that by the corresponding STL, worse otherwise. *Left figures* are for ridge regression(RR) and the right figures are for support vector regression(SVR). **a** Computer dataset. **b** School dataset

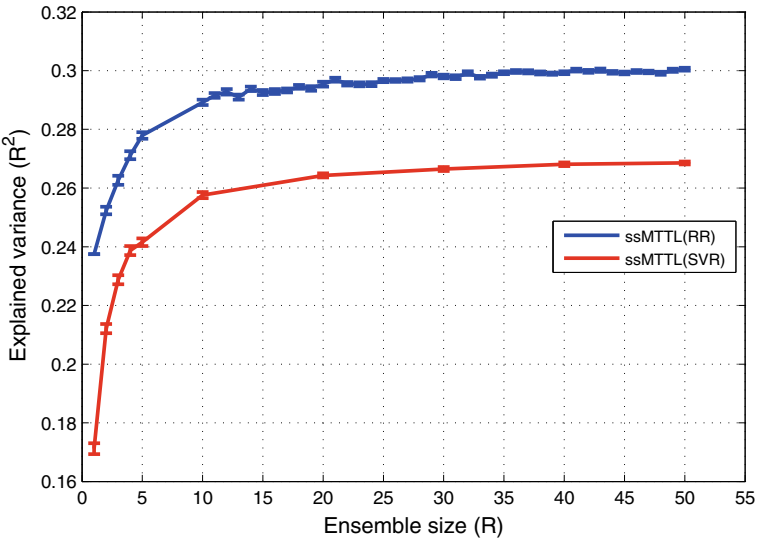

Fig. 14 Explained variance R^2 versus ensemble size (R) for Computer dataset averaged over 50 random training-test splits. Standard errors are reported as error bars

on average in training and 30 students per school for test. For computer dataset, there are 13 binary features with a separate term added to it to account for the bias. We use the ratings of the first 15 computers for training and the remaining for test.

The prediction performance is evaluated in terms of *explained variance* (R^2) and *normalized RMSE* and the results are reported in Table 14. The average performance over 50 random splits are reported with standard error reported in the parenthesis. Table 14a shows the performance of STL, ATL, ssMTTL and three other MTL baselines (GMTL, MTFL, MTRL) when linear regression is used for both the datasets. Average performance over the same splits are shown when Support Vector Regression with RBF kernel is used for all STL, ATL and ssMTTL in Table 14b. It is evident from these tables that ssMTTL over both a linear and a nonlinear regressors resulted in higher performance over the STL and ATL of corresponding regressors. This improvement in performance is consistent in terms of both the evaluation measures. Although the value of R^2 looks small, we note that $R^2 > 0.25$ implies that the correlation between the actual and the predictor is > 0.5, which is a medium level correlation. The Fig. 12 shows the R^2 and the average data per task for ssMTTL with respect to different values of the parameter η. As expected, the performance peaks at certain values of η ($\eta_{optimal} = 0.6$ for school data and $\eta_{optimal} = 0.3$ for computer dataset) for ridge regression (RR), implying that on an average the optimal correction to the task relatedness is achieved at those values of $\eta_{optimal}$. Over-correction of task relatedness with increase in η, as expected, resulted in lower performance.

Figure 13 shows the histogram of task-wise performance for both School and Computer datasets. Tasks are grouped into 3 categories based on whether performance (R^2) by STL and ssMTTL are similar if difference between the R^2 performance, $\Delta R^2 \leq \pm 0.05$, better (if ssMTTL provides at least 0.05 greater R^2 value than that by the corresponding STL), or worse (otherwise). Results are shown for both the ridge regression and support vector regression as the base regressor. It is clearly seen that for both the datasets and for both the base regressors ssMTTL is able to maintain similar or provide better performance for at least 73 % of the tasks.

Figure 14 shows the performance(R^2) as a function of ensemble size (R) for Computer dataset. For both ssMTTL(RR) and ssMTTL(SVR), the performance saturates at $R = 20$ and beyond. This further justifies our choice of $R = 40$ as the ensemble size for ssMTTL. A similar trend is also observed for the School dataset and is therefore omitted.

5 Conclusion

We have presented a *joint modeling* framework that can take any classification or regression algorithm of a practitioner's choice (standard or custom-built) and build its MTL variant. This is achieved by introducing a novel task relatedness measure that can be computed in a way agnostic to the classification or regression technique being used. Tasks are then combined according to their relatedness. As a theoretical basis of our framework, we show that under some mild assumptions, and in the regimen of small training data, the estimates of task parameters are usually poor leading to under-estimation of task relatedness with high probability. Following this result, we derive a MTL algorithm that brings the tasks closer to their "true" relatedness and thus indirectly leads to improved estimates of task parameters. Our algorithm implements MTL variant of a classifier/regressor by appropriately sharing data across tasks, which makes it *agnostic* to the choice of base classification or regression technique. We provide MTL variants for many popular machine learning classifiers/regressors

namely, Logistic regression, SVM, KNN, Random Forest, Ridge regression, Support Vector regression. Our experiments with both synthetic and real datasets clearly demonstrate that the multi-task variants of all these techniques convincingly outperform their single-task learning counterparts. We also show that the proposed model performs comparable or better than many state-of-the-art MTL and transfer learning baselines. Significance of our framework lies in the fact that one can develop multi-task variant of any classifier/regressor technique – standard, custom-built or even yet to be discovered.

The idea proposed in this paper is an initial attempt in building a generic MTL framework. There are several possibilities for improvement. For example, the current way of computing task relatedness crucially depends on data and when sample size is small, these estimates may be poor. Having prior knowledge for task relatedness for a given domain may be helpful. Therefore a possible extension could be to systematically fuse the prior knowledge of task relatedness into data-driven task relatedness in a manner agnostic to a base classifier/regressor. In another extension, it may be possible to replace the ensemble of models by using example-dependent cost sensitive classifiers [23,37,62] where possible. In particular, one can use the probability of including a data point as its weight to the usual cost in such formulations. Replacing ensemble maybe useful toward increasing the computational efficiency of the algorithm. In other future directions that are applicable to privacy-aware MTL, it may be useful to replace our current way of data sharing with some form of privacy-preserving scheme, e.g., data anonymization [12,21] or data perturbation techniques [1].

References

1. Aggarwal CC, Yu PS (2008) A general survey of privacy-preserving data mining models and algorithms. Springer, Berlin
2. Argyriou A, Evgeniou T, Pontil M (2008) Convex multi-task feature learning. Mach Learn 73(3):243–272
3. Baxter J (2000) A model of inductive bias learning. J Artif Intell Res (JAIR) 12:149–198
4. Ben-David S, Schuller R (2003) Exploiting task relatedness for multiple task learning. pp 567–580
5. Bickel S, Brückner M, Scheffer T (2007) Discriminative learning for differing training and test distributions. In: Proceedings of the 24th international conference on machine learning, ACM, pp 81–88
6. Bonilla EV, Chai KM, Williams C (2007) Multi-task Gaussian process prediction. In: Advances in neural information processing systems, pp 153–160
7. Bonilla EV, Agakov FV, Williams C (2007) Kernel multi-task learning using task-specific features. In: International conference on artificial intelligence and statistics, pp 43–50
8. Bonilla EV, Kian CMA, Williams CKI (2007) Multi-task gaussian process prediction. In: Nips, vol 20, pp 153–160
9. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
10. Caruana R (1997) Multitask learning. Mach Learn 28(1):41–75
11. Chen M, Weinberger KQ, Blitzer J (2011) Co-training for domain adaptation. In: NIPS, pp 2456–2464
12. Clifton C, Kantarcioğlu M, Doan A, Schadow G, Vaidya J, Elmagarmid A, Suciu D (2004) Privacy-preserving data integration and sharing. In: Proceedings of the 9th ACM SIGMOD workshop on research issues in data mining and knowledge discovery, ACM, pp 19–26
13. Dai W, Xue G-R, Yang Q, Yu Y (2007) Transferring naive bayes classifiers for text classification. In: Proceedings of the twenty-second AAAI conference on artificial intelligence, vol 22, AAAI Press, p 540
14. Dai W, Yang Q, Xue G-R, Yu Y (2007) Boosting for transfer learning. In: Proceedings of the 24th international conference on machine learning, ACM, pp 193–200
15. Daumé III H (2009) Bayesian multitask learning with latent hierarchies. In: Processing of the 25th conference on uncertainty in artificial intelligence, pp 135–142
16. Daume III H, Marcu D (2006) Domain adaptation for statistical classifiers. J Artif Intell Res, pp 101–126
17. Davis J, Domingos P (2009) Deep transfer via second-order markov logic. In: Proceedings of the 26th annual international conference on machine learning, ACM, pp 217–224
18. Evgeniou A, Pontil M (2007) Multi-task feature learning. In: Advances in neural information processing systems, vol 19, The MIT Press, p 41

19. Evgeniou T, Micchelli CA, Pontil M (2005) Learning multiple tasks with kernel methods. J Mach Learn Res, 615–637
20. Evgeniou T, Pontil M (2004) Regularized multi–task learning. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 109–117
21. Fung BCM, Wang K, Yu PS (2007) Anonymizing classification data for privacy preservation. Knowl Data Eng IEEE Trans 19(5):711–725
22. Gao J, Fan W, Jiang J, Han J (2008) Knowledge transfer via multiple model local structure mapping. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 283–291
23. Geibel P, Brefeld U, Wysotzki F (2003) Learning linear classifiers sensitive to example dependent and noisy costs. In: Advances in intelligent data analysis V, Springer, pp 167–178
24. Gong P, Ye J, Zhang C (2012) Robust multi-task feature learning. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 895–903
25. Gupta SK, Phung D, Adams B, Venkatesh S (2013) Regularized nonnegative shared subspace learning. Data Min Knowl Discov 26(1):57–97
26. Gupta SK, Phung D, Venkatesh S (2012) A slice sampler for restricted hierarchical beta process with applications to shared subspace learning. In: Proceedings of the twenty-eighth conference on uncertainty in artificial intelligence, Catalina Island, CA, USA, 14–18 Aug 2012, pp 316–325
27. Gupta SK, Phung D, Venkatesh S (2013) Factorial multi-task learning: a bayesian nonparametric approach. In: International conference on machine learning, pp 657–665
28. Gupta SK, Rana S, Phung D, Venkatesh S (2015) Collaborating differently on different topics: A multi-relational approach to multi-task learning. In: Advances in knowledge discovery and data mining, Ho Chi Minh City, Vietnam. Springer, Berlin Heidelberg, pp 303–316
29. Gupta SK, Rana S, Phung D, Venkatesh S (2015) What shall I share and with whom? A multi-task learning formulation using multi-faceted task relationships. In: Proceedings of the SIAM international conference on data mining, Vancouver, Canada, pp 703–711
30. Jawanpuria P, Nath JS (2012) A convex feature learning formulation for latent task structure discovery. In: Proceedings of the 29th international conference on machine learning (ICML)
31. Jebara T (2004) Multi-task feature and kernel selection for svms. In: Proceedings of the twenty-first international conference on machine learning, ACM, p 55
32. Kang Z, Grauman K, Sha F (2011) Learning with whom to share in multi-task feature learning. In: Proceedings of the 28th international conference on machine learning, pp 521–528
33. Kumar A, Daumé III H (2012) Learning task grouping and overlap in multi-task learning. In: International conference on machine learning (ICML)
34. Lawrence ND, Platt JC (2004) Learning to learn with the informative vector machine. In: Proceedings of the twenty-first international conference on machine learning, ACM, p 65
35. Lee H, Battle A, Raina R, Ng AY (2006) Efficient sparse coding algorithms. In: Advances in neural information processing systems, pp 801–808
36. Lee S-I, Chatalbashev V, Vickrey D, Koller D (2007) Learning a meta-level prior for feature relevance from multiple related tasks. In: Proceedings of the 24th international conference on machine learning, ACM, pp 489–496
37. Lenarcik A, Piasta Z (1998) Rough classifiers sensitive to costs varying from object to object. In: Rough sets and current trends in computing, Springer, pp 222–230
38. Lenk PJ, De Sarbo WS, Green PE, Young MR (1996) Hierarchical bayes conjoint analysis: recovery of partworth heterogeneity from reduced experimental designs. Mark Sci 15(2):173–191
39. Li S (2011) Concise formulas for the area and volume of a hyperspherical cap. Asian J Math Stat 4(1):66–70
40. Liao X, Xue Y, Carin L (2005) Logistic regression with an auxiliary data source. In: Proceedings of the 22nd international conference on machine learning, ACM, pp 505–512
41. Ling X, Dai W, Xue G-R, Yang Q, Yu Y (2008) Spectral domain-transfer learning. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 488–496
42. Mardia KV, Jupp PE (2009) Directional statistics, vol 494. Wiley, New York
43. McCallum A, Nigam K (1998) A comparison of event models for naive bayes text classification. In: AAAI-98 workshop on learning for text categorization, vol 752, Citeseer, pp 41–48
44. Mihalkova L, Huynh T, Mooney RJ (2007) Mapping and revising markov logic networks for transfer learning. In: AAAI, vol 7, pp 608–614
45. Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359
46. Passos A, Rai P, Wainer J, Daume III H (2012) Flexible modeling of latent task structures in multitask learning. arXiv preprint arXiv:1206.6486

47. Pavlov D, Balasubramanyan R, Dom B, Kapur S, Parikh J (2004) Document preprocessing for naive bayes classification and clustering with mixture of multinomials. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 829–834
48. Pearl J (2012) Some thoughts concerning transfer learning, with applications to meta-analysis and data-sharing estimation. Technical report, Technical Report Technical Report r-387, cognitive systems laboratory, Department of Computer Science, UCLA
49. Platt JC (1999) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Advances in large margin classifiers. Citeseer
50. Raina R, Battle A, Lee H, Packer B, Ng AY (2007) Self-taught learning: transfer learning from unlabeled data. In: proceedings of the 24th international conference on machine learning, ACM, pp 759–766
51. Saha B, Gupta SK, Phung D, Venkatesh S (2014) Multiple task transfer learning with small sample sizes. In: Knowledge and information systems, pp 1–28
52. Shimodaira H (2000) Improving predictive inference under covariate shift by weighting the log-likelihood function. J Stat Plan Inference 90(2):227–244
53. Thrun S (1996) Learning to learn: introduction. In: Learning to learn, Citeseer
54. Van Belle VMCA, Van Calster B, Timmerman D, Bourne T, Bottomley C, Valentin L, Neven P, Van Huffel S, Suykens JAK, Boyd S (2012) A mathematical model for interpretable clinical decision support with applications in gynecology. PloS one 7(3):e34312
55. Wang Q, Zhang L, Chi M, Guo J (2008) MTForest: ensemble decision trees based on multi-task learning. In: European conference on artificial intelligence (ECAI), pp 122–126
56. Wang Z, Song Y, Zhang C (2008) Transferred dimensionality reduction. In: machine learning and knowledge discovery in databases, Springer, pp 550–565
57. Wu P, Dietterich TG (2004) Improving svm accuracy by training on auxiliary data sources. In: Proceedings of the twenty-first international conference on machine learning, ACM, p 110
58. Xue Y, Liao X, Carin L, Krishnapuram B (2007) Multi-task learning for classification with dirichlet process priors. J Mach Learn Res 8:35–63
59. Yang J, Yan R, Hauptmann AG (2007) Cross-domain video concept detection using adaptive svms. In: Proceedings of the 15th international conference on multimedia, pp 188–197
60. Yu K, Tresp V, Schwaighofer A (2005) Learning gaussian processes from multiple tasks. In: Proceedings of the 22nd international conference on Machine learning, ACM, pp 1012–1019
61. Zadrozny B (2004) Learning and evaluating classifiers under sample selection bias. In: Proceedings of the twenty-first international conference on Machine learning, ACM, p 114
62. Zadrozny B, Langford J, Abe N (2003) Cost-sensitive learning by cost-proportionate example weighting. In: Third IEEE international conference on data mining, 2003 (ICDM 2003), IEEE, pp 435–442
63. Zhang Y, Yeung D-Y (2010) A convex formulation for learning task relationships in multi-task learning. In: UAI, pp 733–442
64. Zhou J, Sun J, Liu Y, Hu J, Ye J (2013) Patient risk prediction model via top-k stability selection. In: SIAM conference on data mining. SIAM
65. Zhu J, Chen N, Xing EP (2011) Infinite latent svm for classification and multi-task learning. In: NIPS, pp 1620–1628

Sunil Gupta received this Ph.D. degree from Curtin University in 2012. He is currently a lecturer with the Pattern Recognition and Data Analytics, Deakin University. His research interests include transfer learning, multi-task learning, Bayesian modeling and applications to various data mining applications.

Santu Rana graduated with a Ph.D. in Computing from Curtin University of Technology, Perth, in 2010. He completed his Bachelor in Electrical Engineering from Jadavpur University, Kolkata, in 2003, followed by Masters in System Science and Automation from Indian Institute of Science in 2005. He joined Deakin University Strategic Research Centre of Pattern Recognition and Data Analytics (PRaDA) as a lecturer from the centres inception in 2012. He is an expert in multi-linear modeling of big data, Bayesian nonparametric modeling of dynamic system, and healthcare data analysis.

Budhaditya Saha received this Ph.D. degree from Curtin University in 2010. He was a post-doctoral researcher with the Institute for Multi-sensor Processing and Content Analysis, Curtin University. He is currently a lecturer with the Pattern Recognition and Data Analytics, Deakin University. His research interest is mainly on sparse modeling, anomaly detection and large-scale data mining applications.

Dinh Phung received the Ph.D. degree in Computer Science from Curtin University in 2005. He is currently a Associate Professor at school of Information Technology at Deakin University, Melbourne, Victoria. His research interests include Bayesian machine methods and their applications to multimedia and social networks analysis.

Svetha Venkatesh is Alfred Deakin Distinguished Professor at school of Information Technology at Deakin University, Melbourne, Victoria. Venkatesh has extensive experience in low-level vision, pattern recognition and multimedia content analysis. Her research is in the areas of large scale pattern recognition, image understanding, multimedia analysis and applications of computer vision to image and video indexing and retrieval. She is the author of about 320 research papers in these areas and is currently the Director of the Institute of Multi-sensor Processing and Content Analysis. She is also the Associate Editor of IEEE Transaction in Multimedia, and ACM Transactions on Multimedia Computing, Communications and Applications (2008 onwards). She has been awarded the Fellowship of the International Association of Pattern Recognition and Fellow of Australian Academy of Technological Science and Engineering.