Multi-Task Boosting by Exploiting Task Relationships

Yu Zhang and Dit-Yan Yeung

Hong Kong University of Science and Technology
{zhangyu,dyyeung}@cse.ust.hk

Abstract. Multi-task learning aims at improving the performance of one learning task with the help of other related tasks. It is particularly useful when each task has very limited labeled data. A central issue in multi-task learning is to learn and exploit the relationships between tasks. In this paper, we generalize boosting to the multi-task learning setting and propose a method called multi-task boosting (MTBoost). Different tasks in MTBoost share the same base learners but with different weights which are related to the estimated task relationships in each iteration. In MTBoost, unlike ordinary boosting methods, the base learners, weights and task covariances are learned together in an integrated fashion using an alternating optimization procedure. We conduct theoretical analysis on the convergence of MTBoost and also empirical analysis comparing it with several related methods.

1 Introduction

In many real-world applications, the amount of labeled data available in a single learning task is scarce but there exist multiple related tasks. Multi-task learning [1–3] exploits this scenario to improve the performance of one learning task with the help of other related tasks. This learning paradigm, which can date back to some research in psychology and cognitive science, is inspired by human learning ability in that people often apply the knowledge gained from previous learning tasks to help learn a new task. For example, if a person can play Chinese chess, then (s)he will learn to play chess more easily by transferring the knowledge gained from playing Chinese chess. Major advances have been made in multi-task learning over the past decade. Multi-layered feedforward neural network [1] is one of the earliest models for multi-task learning. The units of the hidden layer in a neural network represent the common features for data points from all tasks and each unit in the output layer usually corresponds to the output of one task. Besides multi-layered feedforward neural networks, multi-task feature learning [4, 5] also learns common features for all tasks with the difference being that it is a regularized method. Different from these methods which learn common data representations, some methods aim to learn similar model parameters for different tasks, e.g., regularized multi-task support vector machine (SVM) [6] defines a new regularizer to enforce the SVM parameters for all tasks to be close to each other. Moreover, one widely used approach for multi-task learning is the task clustering approach [7–9] in which the main idea is to group the tasks into several clusters and then learn identical or similar data features or model parameters for the tasks within each cluster. An advantage of this approach over the above mentioned methods is its robustness against outlier

tasks because they reside in separate clusters that do not affect other tasks. Among many existing methods proposed for multi-task learning, a central issue in multi-task learning is to learn and exploit the pairwise relationships between tasks. There are three types of pairwise task relationships, namely, positive task correlation, negative task correlation, and task unrelatedness. However, most existing multi-task learning methods cannot make full use of all three types of task relationships. One way to incorporate the task relationships into a learning model is by adopting some model assumption about task relatedness. Unfortunately, the model assumption adopted may be incorrect. Worse still, it is not easy to verify the correctness of the model assumption. As such, it is more desirable to take an alternative approach by learning the task relationships from data automatically. The multi-task Gaussian process (GP) model [10] and its extension [11] are recently proposed methods that adopt this approach under the Bayesian framework. Moreover, Zhang and Yeung proposed a method in [12] to learn task relationships under the regularization framework for classification and regression problems, and then extended it for feature selection problems in [13].

Boosting [14], which seeks to combine multiple (weak) base learners to form a learner with significantly better performance, has been widely used in many areas, such as machine learning and data mining. There exist some explanations, e.g., margin theory [15, 16], for the success of boosting. Moreover, some studies show that boosting is related to the additive model [17, 18] in statistics. Even though boosting methods have shown good performance in many applications, their performance is often unsatisfactory when the labeled data is scarce. This calls for combining boosting and multi-task learning [19, 20].

In this paper, we generalize boosting to the multi-task learning setting via learning and exploiting the pairwise task relationships. Our point of departure is a regularized method in [12] which learns the task relationships in the form of a task covariance matrix under a regularization framework and is related to maximum a posteriori (MAP) estimation of the weight-space interpretation of the multi-task GP model [10] presented in [11]. We then extend the formulation for boosting to give a method called multi-task boosting (MTBoost). By viewing boosting as a feature generating process, different tasks in MTBoost share the same base learners but with different weights which are related to the estimated task relationships in each iteration. Unlike single-task boosting methods which learn the base learners and weights separately, the base learners, weights and task covariances in MTBoost are learned together in an integrated fashion using an alternating optimization procedure. We conduct theoretical analysis on the convergence of the MTBoost learning algorithm.

The remainder of this paper is organized as follows. We present our MTBoost model and its learning algorithm in Section 2. Section 3 reviews some related work and Section 4 reports experimental results on some multi-task learning applications. Concluding remarks are given in the final section.

2 Multi-Task Boosting by Exploiting Task Relationships

Let there be m learning tasks $\{T_i\}_{i=1}^m$. For the ith task T_i, the training set consists of n_i labeled data points $(\mathbf{x}_j^i, y_j^i)$, $j = 1, \ldots, n_i$, with $\mathbf{x}_j^i \in \mathbb{R}^d$ and its corresponding output $y_j^i \in \{-1, 1\}$ for a binary classification problem.

2.1 Task Covariance Matrix

In [10], Bonilla et al. proposed a multi-task GP model which models the pairwise task relationships using a task covariance matrix under the Bayesian framework. However, it is not clear how to learn the task covariance matrix for other models such as those formulated under the regularization framework. In [12], the authors presented a regularized method, shedding light on how the task covariance matrix affects the learning of multiple tasks. More specifically, the task covariance matrix $\boldsymbol{\Omega}$ is used as a parameter matrix for the matrix-variate normal distribution [21] over the model parameters in least squares regression or support vector machine (SVM):

$$\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_m) \sim \mathcal{MN}_{d\times m}(\mathbf{A} \,|\, \mathbf{0}_{d\times m}, \mathbf{I}_d \otimes \boldsymbol{\Omega}), \tag{1}$$

where $\mathbf{a}_i \in \mathbb{R}^d$ is the model parameter vector for the ith task, $\mathcal{MN}_{d\times m}(\mathbf{M}, \boldsymbol{\Sigma} \otimes \boldsymbol{\Omega})$ denotes a matrix-variate normal distribution with mean $\mathbf{M} \in \mathbb{R}^{d\times m}$, row covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ and column covariance matrix $\boldsymbol{\Omega} \in \mathbb{R}^{m\times m}$. The probability density function of the matrix-variate normal distribution is

$$p(\mathbf{X} \,|\, \mathbf{M}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}) = \frac{\exp\left(-\frac{1}{2}\text{tr}\left(\boldsymbol{\Sigma}^{-1}(\mathbf{X}-\mathbf{M})\boldsymbol{\Omega}^{-1}(\mathbf{X}-\mathbf{M})^T\right)\right)}{(2\pi)^{md/2}|\boldsymbol{\Sigma}|^{m/2}|\boldsymbol{\Omega}|^{d/2}},$$

where $\text{tr}(\cdot)$ denotes the trace of a square matrix, $|\cdot|$ denotes the determinant of a square matrix, and $\mathbf{B}^{-1}$ denotes the inverse of a non-singular matrix $\mathbf{B}$ or the pseudo-inverse when it is singular. From this view, we can see that $\boldsymbol{\Omega}$ is used to model the covariance between the columns in the model parameter matrix $\mathbf{A}$ and hence to model the task relationships since each column in $\mathbf{A}$ represents the model parameters of the corresponding task.

Given the likelihood (i.e., logistic model for classification problem or Gaussian noise model for regression problem) and the prior defined in Eq. (1), the MAP solution is obtained by solving the following problem:

$$\begin{aligned} \min_{\mathbf{A},\boldsymbol{\Omega}} \; & \sum_{i=1}^{m}\sum_{j=1}^{n_i} l(\mathbf{a}_i^T\mathbf{x}_j^i, y_j^i) + \frac{\lambda}{2}\text{tr}(\mathbf{A}\boldsymbol{\Omega}^{-1}\mathbf{A}^T) \\ \text{s.t.} \; & \boldsymbol{\Omega} \succeq 0, \; \text{tr}(\boldsymbol{\Omega}) = 1, \end{aligned} \tag{2}$$

where $l(\cdot,\cdot)$ defines the empirical loss corresponding to the likelihood, λ is a regularization parameter which balances the tradeoff between the empirical loss and the regularization term, and $\boldsymbol{\Omega} \succeq 0$ means that the matrix $\boldsymbol{\Omega}$ is positive semidefinite. The first constraint in problem (2) is needed because $\boldsymbol{\Omega}$ is defined as a task covariance matrix and the second constraint serves to restrict the complexity of $\boldsymbol{\Omega}$. The second term

in the objective function of problem (2) is derived from the matrix-variate normal prior in Eq. (1) and is used to regularize the task relationships.

It is easy to show that problem (2) is a convex problem as long as the loss function $l(\cdot,\cdot)$ is convex, as proved in [12]. We will show how to design a boosting algorithm according to problem (2).

2.2 Multi-Task Boosting

In a boosting algorithm, we are given a fixed class of functions (or called base hypotheses) denoted by $\mathcal{F}$ and are required to find a linear combination of functions in $\mathcal{F}$, denoted by $\text{lin}(\mathcal{F})$, that minimizes a cost functional $C(\cdot)$ on $\text{lin}(\mathcal{F})$. For example, in our experiments, due to the high-dimensional data involved, a linear least-squares SVM is utilized as the base learner. The final hypothesis can be written as $F(\mathbf{x}) = \sum_{t=1}^{Q} w_t f_t(\mathbf{x})$ where $f_t \in \mathcal{F}$ and $w_i \in \mathbb{R}$. So, boosting may be viewed as finding for each data point $\mathbf{x}$ a new feature representation $\mathbf{z} = (f_1(\mathbf{x}), \ldots, f_Q(\mathbf{x}))^T$ and also a coefficient vector. From this view, we can extend problem (2) for multi-task boosting as

$$\begin{aligned} \min_{\mathbf{W},\boldsymbol{\Omega},\{f_t\}} \ & \sum_{i=1}^{m}\sum_{j=1}^{n_i} l(\mathbf{w}_i^T\mathbf{z}_j^i, y_j^i) + \frac{\lambda}{2}\text{tr}(\mathbf{W}\boldsymbol{\Omega}^{-1}\mathbf{W}^T) \\ \text{s.t.} \ & \mathbf{z}_j^i = (f_1(\mathbf{x}_j^i), \ldots, f_Q(\mathbf{x}_j^i))^T \ \forall i,j \\ & \boldsymbol{\Omega} \succeq 0, \ \text{tr}(\boldsymbol{\Omega}) = 1, \end{aligned} \tag{3}$$

where $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_m)$. In this formulation, we can see that the final hypothesis F_i for the ith task can be expressed as $F_i = \sum_{j=1}^{Q} w_{ij} f_j$, where w_{ij} is the jth element of $\mathbf{w}_i$. So different tasks share the same base hypotheses but with different weights.

However, here we cannot know the base hypotheses in advance. So we view this model as an additive model [17] and use the gradient boosting technique [18, 22] to learn the base hypotheses and their weights. More specifically, in the t-th iteration, the existing combined hypothesis for the ith task is denoted by $F_i^{(t-1)}$ and the optimization problem is

$$\begin{aligned} \min_{\mathbf{w}^t,\boldsymbol{\Omega},f_t} \ & \sum_{i=1}^{m} C(F_i^{(t-1)} + w_{it} f_t) + \frac{\lambda}{2}\text{tr}(\mathbf{W}_t\boldsymbol{\Omega}^{-1}\mathbf{W}_t^T) \\ \text{s.t.} \ & \boldsymbol{\Omega} \succeq 0, \ \text{tr}(\boldsymbol{\Omega}) = 1, \end{aligned} \tag{4}$$

where $\mathbf{W}_t$ is the weight matrix until the tth iteration with (i,j)th element as w_{ji} and $\mathbf{w}^t = (w_{1t}, \ldots, w_{mt})$ denotes the new weight vector obtained in the tth iteration. Here $C(F_i) = \sum_{j=1}^{n_i} l(F_i(\mathbf{x}_j^i), y_j^i)$. Since we mainly consider the classification problem in this section, we take the margin cost functional to be the loss function, i.e., $l(F(\mathbf{x}_j^i), y_j^i) = c(y_j^i F(\mathbf{x}_j^i))$ for some monotonically decreasing function $c(\cdot)$. In this paper, $c(\cdot)$ takes the form of $c(x) = \ln(1+\exp(-x))$ which is widely used in boosting algorithms such as LogitBoost [17].

Since problem (4) is still not easy to solve, we use the majorization-minimization (MM) algorithm [23] to solve it. The MM algorithm is an iterative algorithm which seeks an

upper bound of the objective function based on the solution of the previous iteration as a surrogate function for a minimization problem and minimizes the surrogate function instead of the original objective function. It has been proved that the MM algorithm is guaranteed to find a local optimum. Here for simplicity, we just run one iteration of the MM algorithm with the initial solution of f_t as a zero function. Since $c(x) = \ln(1+\exp(-x))$ is a concave function due to the fact that $\frac{\partial^2 c(x)}{\partial^2 x} = -\frac{\exp(-x)}{(1+\exp(-x))^2} < 0$, $C(\cdot)$ is also a concave functional and hence we have

$$C(F_i^{(t-1)} + w_{it} f_t) \leq C(F_i^{(t-1)}) + w_{it}\langle \nabla C(F_i^{(t-1)}), f_t \rangle,$$

due to the first-order property of a concave function. Here $\nabla C(F_i)$ denotes the functional derivative of C at F_i and $\langle F_i, G_i \rangle$ is the inner product which is defined as $\langle F_i, G_i \rangle = \sum_{j=1}^{n_i} F_i(\mathbf{x}_j^i) G_i(\mathbf{x}_j^i)$. So in each iteration of the MM algorithm, the optimization problem can be formulated as

$$\begin{aligned} \min_{\mathbf{w}^t, \boldsymbol{\Omega}, f_t} \ & \sum_{i=1}^{m} w_{it} \langle \nabla C(F_i^{(t-1)}), f_t \rangle + \frac{\lambda}{2} \mathrm{tr}(\mathbf{W}_t \boldsymbol{\Omega}^{-1} \mathbf{W}_t^T) \\ \text{s.t.} \ & \boldsymbol{\Omega} \succeq 0, \ \mathrm{tr}(\boldsymbol{\Omega}) = 1. \end{aligned} \tag{5}$$

Unlike conventional boosting algorithms which can optimize w_{it} and f_t separately as in [18, 22], here in problem (5) w_{it} and f_t are coupled together. We use an alternating method to solve the problem.

When $\mathbf{w}^t$ and $\boldsymbol{\Omega}$ are given, we can get

$$\langle \nabla C(F_i^{(t-1)}), f_t \rangle = \sum_{j=1}^{n_i} y_j^i f_t(\mathbf{x}_j^i) c'(y_j^i F_i^{(t-1)}(\mathbf{x}_j^i)),$$

where $c'(\cdot)$ is the derivative of $c(\cdot)$, since $C(F_i^{(t-1)}) = \sum_{j=1}^{n_i} c(y_j^i F_i^{(t-1)}(\mathbf{x}_j^i))$. Then we need to solve the following minimization problem to find f_t:

$$\min_{f_t} \sum_{i=1}^{m} w_{it} \sum_{j=1}^{n_i} y_j^i f_t(\mathbf{x}_j^i) c'(y_j^i F_i^{(t-1)}(\mathbf{x}_j^i)). \tag{6}$$

Since $c(\cdot)$ is monotonically decreasing, the derivative $c'(\cdot)$ is negative. Problem (6) can be reformulated as

$$\max_{f_t} \sum_{i=1}^{m} \sum_{j=1}^{n_i} \tilde{y}_j^i f_t(\mathbf{x}_j^i) d_j^i, \tag{7}$$

where $d_j^i = \frac{c'(y_j^i F_i^{(t-1)}(\mathbf{x}_j^i))|w_{it}|}{\sum_{i=1}^{m} |w_{it}| \sum_{j=1}^{n_i} c'(y_j^i F_i^{(t-1)}(\mathbf{x}_j^i))}$ defines the instance weight for $\mathbf{x}_j^i$, $\tilde{y}_j^i = \mathrm{sign}(w_{it}) y_j^i$, and $\mathrm{sign}(\cdot)$ is the sign function. Since $w_{it} \in \mathbb{R}$, here we take the absolute value of w_{it} to keep the instance weights $\{d_j^i\}$ non-negative and the equivalence between problem (6) and (7) is due to the fact that $w_{it} = \mathrm{sign}(w_{it})|w_{it}|$. We assume $f_t(\cdot) \in \{-1, 1\}$. Since $\tilde{y}_j^i \in \{-1, 1\}$,[1] the objective function in problem (7) can be

[1] When $w_{it} = 0$, the ith task has no effect on problem (7) and hence can be ignored.

rewritten as

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i}\tilde{y}_j^i f_t(\mathbf{x}_j^i)d_j^i = \sum_{\tilde{y}_j^i=f_t(\mathbf{x}_j^i)} d_j^i - \sum_{\tilde{y}_j^i\neq f_t(\mathbf{x}_j^i)} d_j^i = 1 - 2\sum_{\tilde{y}_j^i\neq f_t(\mathbf{x}_j^i)} d_j^i.$$

So problem (7) is equivalent to minimizing the weighted classification error

$$\min_{f_t} \sum_{\tilde{y}_j^i\neq f_t(\mathbf{x}_j^i)} d_j^i. \tag{8}$$

From problem (8), we may look at it as a weighted combination of multiple tasks to give a single "supertask" with possible label flipping from y_j^i to $\tilde{y}_j^i$ depending on the relationships between tasks. For a base learner such as least-squares SVM, we need to solve a weighted least-squares SVM where the instance weights are defined by $\{d_j^i\}$.

When f_t and $\mathbf{w}^t$ are given, we need to solve the following problem

$$\begin{aligned} &\min_{\boldsymbol{\Omega}} \ \text{tr}(\mathbf{W}_t\boldsymbol{\Omega}^{-1}\mathbf{W}_t^T) \\ &\text{s.t.} \ \ \boldsymbol{\Omega}\succeq 0,\ \text{tr}(\boldsymbol{\Omega}) = 1. \end{aligned} \tag{9}$$

Then we have

$$\begin{aligned} \text{tr}(\boldsymbol{\Omega}^{-1}\mathbf{B}) &= \text{tr}(\boldsymbol{\Omega}^{-1}\mathbf{B})\text{tr}(\boldsymbol{\Omega}) \\ &= \text{tr}((\boldsymbol{\Omega}^{-\frac{1}{2}}\mathbf{B}^{\frac{1}{2}})(\mathbf{B}^{\frac{1}{2}}\boldsymbol{\Omega}^{-\frac{1}{2}}))\text{tr}(\boldsymbol{\Omega}^{\frac{1}{2}}\boldsymbol{\Omega}^{\frac{1}{2}}) \\ &\geq (\text{tr}(\boldsymbol{\Omega}^{-\frac{1}{2}}\mathbf{B}^{\frac{1}{2}}\boldsymbol{\Omega}^{\frac{1}{2}}))^2 = (\text{tr}(\mathbf{B}^{\frac{1}{2}}))^2, \end{aligned}$$

where $\mathbf{B} = \mathbf{W}_t^T\mathbf{W}_t$. The first equality holds because of the last constraint in problem (9), and the last inequality holds because of the Cauchy-Schwarz inequality for the Frobenius norm. Moreover, $\text{tr}(\boldsymbol{\Omega}^{-1}\mathbf{B})$ attains its minimum value $(\text{tr}(\mathbf{B}^{\frac{1}{2}}))^2$ if and only if $\boldsymbol{\Omega}^{-\frac{1}{2}}\mathbf{B}^{\frac{1}{2}} = a\boldsymbol{\Omega}^{\frac{1}{2}}$ for some constant a and $\text{tr}(\boldsymbol{\Omega}) = 1$. So we can get the analytical solution

$$\boldsymbol{\Omega} = \frac{(\mathbf{W}_t^T\mathbf{W}_t)^{\frac{1}{2}}}{\text{tr}\left((\mathbf{W}_t^T\mathbf{W}_t)^{\frac{1}{2}}\right)}. \tag{10}$$

When f_t and $\boldsymbol{\Omega}$ are given, the optimization problem for $\mathbf{w}^t$ is formulated as

$$\min_{\mathbf{w}^t} J(\mathbf{w}^t) = \mathbf{w}^t\boldsymbol{\beta}_t + \frac{\lambda}{2}\mathbf{w}^t\boldsymbol{\Omega}^{-1}(\mathbf{w}^t)^T, \tag{11}$$

where $\boldsymbol{\beta}_t = (\langle\nabla C(F_1^{(t-1)}), f_t\rangle, \ldots, \langle\nabla C(F_m^{(t-1)}), f_t\rangle)^T$. We set the derivative of problem (11) with respect to $\mathbf{w}^t$ to zero to get the solution of $\mathbf{w}^t$ as

$$\mathbf{w}^t = -\frac{1}{\lambda}(\boldsymbol{\beta}_t)^T\boldsymbol{\Omega}. \tag{12}$$

We summarize the MTBoost algorithm in Table 1.

Table 1. Algorithm for Multi-Task Boosting (MTBoost)

Input: $\{(\mathbf{x}_j^i, y_j^i)\}_{j=1}^{n_i}$ $(i = 1, \ldots, m)$, λ, maximum numbers of iterations Q and Q_1
Let $F_i^{(0)}(x) := 0$ for $i = 1, \ldots, m$ and $\boldsymbol{\Omega} := \mathbf{I}_m$;
for $t := 1$ to Q do
Initialize $\mathbf{w}^t$;
for $t_1 := 1$ to Q_1
Update f_t by solving problem (8);
Update $\boldsymbol{\Omega}$ via Eq. (10);
Update $\mathbf{w}^t$ via Eq. (12);
end for
Let $F_i^{(t)} := F_i^{(t-1)} + w_{it} f_t$ for $i = 1, \ldots, m$;
if $(\boldsymbol{\beta}_t)^T \boldsymbol{\Omega} \boldsymbol{\beta}_t \leq \varepsilon$
break;
end if
end for
Output: $F_i^{(Q)}$ for $i = 1, \ldots, m$

For the initialization of $\mathbf{w}^t$, we randomly generate it from a normal distribution with zero mean and $\boldsymbol{\Omega}$ as the covariance matrix due to the matrix-variate normal prior on $\mathbf{W}_t$ in Eq. (1).

The whole procedure of our MTBoost algorithm includes solving problem (8) and updating $\boldsymbol{\Omega}$ and $\mathbf{w}^t$ according to Eqs (10) and (12). The main computational cost lies in solving problem (8) whose complexity equals the computational cost of training a base learner (i.e., SVM or least-squares SVM) on the training data of all tasks.

2.3 Theoretical Analysis

In this section, we prove the convergence of the MTBoost algorithm.

Theorem 1. *The solution minimizing problem (5) also minimizes problem (4).*

Proof Let $G(\mathbf{w}^t, \boldsymbol{\Omega}, f_t)$ denote the objective function of problem (4) and $H(\mathbf{w}^t, \boldsymbol{\Omega}, f_t)$ denote the objective function of problem (5). Due to the concavity of $C(\cdot)$, for any $\mathbf{w}^t$, $\boldsymbol{\Omega}$ and f_t, we have

$$G(\mathbf{w}^t, \boldsymbol{\Omega}, f_t) \leq H(\mathbf{w}^t, \boldsymbol{\Omega}, f_t) + \sum_{i=1}^{m} C(F_i^{(t-1)}).$$

Moreover, we have

$$G(\mathbf{w}_0^t, \boldsymbol{\Omega}_0, 0) = H(\mathbf{w}_0^t, \boldsymbol{\Omega}_0, 0) + \sum_{i=1}^{m} C(F_i^{(t-1)}),$$

where 0 denotes the zero function and $\mathbf{w}_0^t$ and $\boldsymbol{\Omega}_0$ are the initial values for the variables $\mathbf{w}^t$ and $\boldsymbol{\Omega}$. For the solution $(\mathbf{w}^t, \boldsymbol{\Omega}, f_t)$ minimizing problem (5), we have

$$H(\mathbf{w}^t, \boldsymbol{\Omega}, f_t) \leq H(\mathbf{w}_0^t, \boldsymbol{\Omega}_0, 0).$$

Then we can get

$$\begin{aligned} G(\mathbf{w}^t, \boldsymbol{\Omega}, f_t) &\leq H(\mathbf{w}^t, \boldsymbol{\Omega}, f_t) + \sum_{i=1}^{m} C(F_i^{(t-1)}) \\ &\leq H(\mathbf{w}_0^t, \boldsymbol{\Omega}_0, 0) + \sum_{i=1}^{m} C(F_i^{(t-1)}) \\ &= G(\mathbf{w}_0^t, \boldsymbol{\Omega}_0, 0), \end{aligned}$$

which means the value of the objective function of problem (4) at the solution of problem (5) is lower than that at the initial values. Hence we prove the result. □

Theorem 2.

$$\sum_{i=1}^{m} C(F_i^{(t)}) \leq \sum_{i=1}^{m} C(F_i^{(t-1)}) - \frac{1}{\lambda}\boldsymbol{\beta}_t \boldsymbol{\Omega} \boldsymbol{\beta}_t \leq \sum_{i=1}^{m} C(F_i^{(t-1)})$$

Proof Due to the concavity of $C(\cdot)$, we have

$$\begin{aligned} C(F_i^{(t)}) &= C(F_i^{(t-1)} + w_{it} f_t) \\ &\leq C(F_i^{(t-1)}) + w_{it} \langle \nabla C(F_i^{(t-1)}), f_t \rangle. \end{aligned}$$

Then we can get

$$\begin{aligned} \sum_{i=1}^{m} C(F_i^{(t)}) \leq & \sum_{i=1}^{m} C(F_i^{(t-1)}) + w_{it} \langle \nabla C(F_i^{(t-1)}), f_t \rangle \\ = & \sum_{i=1}^{m} C(F_i^{(t-1)}) + \mathbf{w}^t \boldsymbol{\beta}_t \\ = & \sum_{i=1}^{m} C(F_i^{(t-1)}) - \frac{1}{\lambda} (\boldsymbol{\beta}_t)^T \boldsymbol{\Omega} \boldsymbol{\beta}_t, \end{aligned}$$

where the last equality holds due to the relationship between $\mathbf{w}^t$ and $\boldsymbol{\beta}_t$ reflected in Eq. (12). Moreover, since $\boldsymbol{\Omega}$ is a positive semi-definite matrix which can be verified by the solution of $\boldsymbol{\Omega}$ in Eq. (10), we have

$$\frac{1}{\lambda} (\boldsymbol{\beta}_t)^T \boldsymbol{\Omega} \boldsymbol{\beta}_t \geq 0$$

and hence

$$\sum_{i=1}^{m} C(F_i^{(t-1)}) - \frac{1}{\lambda}\boldsymbol{\beta}_t \boldsymbol{\Omega} \boldsymbol{\beta}_t \leq \sum_{i=1}^{m} C(F_i^{(t-1)}).$$

Finally we reach the conclusion. □

From Theorem 2, we can see that when adding a new component classifier in MTBoost, the empirical loss of all tasks decreases. Since the empirical loss is non-negative, our method is guaranteed to converge. Moreover, Theorem 2 suggests a termination criterion for the MTBoost algorithm in Table 1: $(\boldsymbol{\beta}_t)^T \boldsymbol{\Omega} \boldsymbol{\beta}_t$ is small and below a threshold ε.

3 Related Work

Duchi and Singer [24] proposed a boosting method for multi-class classification problems by utilizing the structural sparsity of model parameters. They claimed that the method can be generalized for multi-task learning. An underlying assumption of their method is that all tasks are similar and they share a similar model or data representation. However, in many applications, there exist tasks which exhibit negative task correlation or task unrelatedness and hence the assumption is violated, impairing the performance of the method.

Wang et al. [19] extended the idea of task clustering [8] to boosting by grouping the tasks into several clusters and learning similar data features or model parameters for the tasks within each cluster. This approach is robust against outlier tasks because outlier tasks reside in separate clusters that do not affect other tasks, but they are local methods in the sense that only similar tasks within the same task cluster can interact to help each other, thus ignoring negative task correlation which may exist between tasks residing in different clusters. Moreover, how to determine the number of clusters is a difficult model selection problem.

Chapelle et al. [20] proposed a multi-boost method for multi-task learning which assumes that the model parameters in different tasks are similar and utilizes the difference between different model parameters to define a regularization term for boosting. The relationship between the multi-boost method and single-task boosting is similar to that between regularized multi-task SVM [6] and single-task SVM. Moreover, similar to [24], the multi-boost method also uses l_1 regularization to enforce sparsity.

Dai et al. [25] proposed a boosting method for transfer learning. Transfer learning is related to multi-task learning but there exist some differences. The tasks in transfer learning can be divided into source and target tasks and transfer learning aims at improving the performance of the target task with the help of the source tasks while multi-task learning seeks to improve the performance of all tasks simultaneously. Pardoe and Stone [26] extended boosting to regression problems in the transfer setting.

4 Experiment

In this section, we study MTBoost empirically on some applications and compare it with a single-task boosting method called AnyBoost [22], a multi-task boosting method called Multi-boost [20], a multi-task learning method called multi-task GP (MTGP) [10] and MTRL [12] which can also learn the task relationships under the GP and regularization framework respectively.[2]

4.1 Multi-Domain Sentiment Classification

In this subsection, we study a multi-domain sentiment classification application[3] which is naturally a multi-task classification problem. Its goal is to classify the reviews of

[2] The implementation of our method can be downloaded from `http://www.cse.ust.hk/~dyyeung/code/MTBoost.zip`.

[3] http://www.cs.jhu.edu/∼mdredze/datasets/sentiment/

some products into two classes: positive and negative reviews. In this application, there are four different products (tasks) from Amazon.com: books, DVDs, electronics, and kitchen appliances. For each task, there are 1,000 positive and 1,000 negative data points corresponding to positive and negative reviews, respectively. Each data point has 473,856 feature dimensions.

Since the feature dimensionality is very high making tree classifiers such as C4.5 and decision stump difficult to use, we choose linear least-squares SVM as the base learner in AnyBoost, Multi-boost and our MTBoost method. To simulate real applications in which the labeled data is scarce, we choose only 20% of the data in each task to form the training set and the rest to form the test set. We perform 10 random splits of the data and report the mean and standard deviation over the 10 trials. The number of rounds in AnyBoost, Multi-Boost and MTBoost is set to 100 and the number of the inner iterations of our MTBoost (i.e., Q_1) is set to 10. The optimal λ is determined by 5-fold cross validation where the candidate set is $\{0.01, 0.1, 1, 10, 100\}$. The results are summarized in Table 2 and the best result after pairwise t-test is shown in bold. From the table, we can see that MTBoost outperforms AnyBoost, Multi-boost, MTGP and MTRL on almost every task. Moreover, we notice that the performance of Multi-boost is just comparable or even worse than that of AnyBoost. One possible reason is that not all the tasks are very similar to each other as can be revealed by the mean task correlation matrix shown in Table 3. In other words, the assumption underlying Multi-boost is not satisfied well in this data set.

Table 2. Comparison of different methods on multi-domain sentiment classification. Each column in the table represents one task. For each method, the first row records the mean classification error over 10 trials and the second row records the standard deviation. 1st task: books; 2nd task: DVDs; 3rd task: electronics; 4th task: kitchen appliances.

Method	1st Task	2nd Task	3rd Task	4th Task
AnyBoost	0.2595	0.2500	0.1999	0.1789
	0.0086	0.0085	0.0096	0.0054
MTGP	0.2594	0.2510	0.2493	0.2407
	0.0097	0.0089	0.0076	0.0085
Multi-boost	0.2918	0.3041	0.3116	0.3165
	0.0138	0.0122	0.0204	0.0175
MTRL	0.2474	**0.2233**	0.1925	0.1719
	0.0116	0.0115	0.0135	0.0098
MTBoost	**0.2385**	**0.2236**	**0.1666**	**0.1491**
	0.0105	0.0087	0.0054	0.0114

The mean task correlation matrix over 10 trials is shown in Table 3. We can see that the first task 'books' is more correlated with the second task 'DVDs' than with the other tasks; the third and fourth tasks achieve the highest correlation among all pairs of tasks. The finding from Table 3 about the relationships between tasks matches our intuition,

with the following possible interpretation: 'books' and 'DVDs' are mainly for entertainment; and almost all the elements in 'kitchen appliances' belong to 'electronics'.

Table 3. Mean task correlation matrix over 10 trials for multi-domain sentiment data. 1st task: books; 2nd task: DVDs; 3rd task: electronics; 4th task: kitchen appliances.

	1st	2nd	3rd	4th
1st	1.0000	0.6977	0.6253	0.6357
2nd	0.6977	1.0000	0.6306	0.6186
3rd	0.6253	0.6306	1.0000	0.7994
4th	0.6357	0.6186	0.7994	1.0000

Table 4. Comparison of different methods on handwritten letter classification. Each column in the table represents one task. For each method, the first row records the mean classification error over 10 trials and the second row records the standard deviation.

Method	1st Task	2nd Task	3rd Task	4th Task	5th Task	6th Task	7th Task
AnyBoost	0.1330	0.3026	0.1271	0.0970	0.0895	0.1997	0.0689
	0.0231	0.0135	0.0311	0.0068	0.0137	0.0178	0.0062
MTGP	0.1316	0.2844	0.1146	0.0903	0.1349	0.2177	0.0852
	0.0135	0.0070	0.0153	0.0075	0.0256	0.0388	0.0456
Multi-boost	0.2064	0.3425	0.2144	0.1295	0.1301	0.2111	0.0989
	0.0548	0.1238	0.1001	0.0675	0.0175	0.0327	0.0754
MTRL	0.1184	0.2790	0.1058	0.0824	0.0880	0.2180	**0.0561**
	0.0048	0.0128	0.0090	0.0060	0.0057	0.0131	0.0073
MTBoost	**0.1136**	**0.2642**	**0.1001**	**0.0611**	**0.0830**	**0.1924**	0.0623
	0.0161	0.0108	0.0127	0.0066	0.0105	0.0233	0.0078

4.2 Handwritten Letter Classification

The handwritten letter classification applicaton[4] consists of seven tasks each of which is a binary classification problem. The corresponding letter pairs for the seven tasks are: c/e, g/y, m/n, a/g, a/o, f/t and h/n. Each data point has 128 features corresponding to the pixel values of the handwritten letter images. For each task, there are about 1,000 positive and 1,000 negative data points. The experimental settings are the same as those for the multi-domain sentiment application above.

The mean classification errors and the standard deviations of different methods over 10 trials are summarized in Table 4. The results show that MTBoost outperforms AnyBoost on every task, showing the effectiveness of sharing between multiple learning

[4] http://multitask.cs.berkeley.edu/

Table 5. Comparison of different methods on USPS digit classification. Each column in the table represents one task. For each method, the first row records the mean classification error over 10 trials and the second row records the standard deviation.

Method	1st Task	2nd Task	3rd Task	4th Task	5th Task	6th Task	7th Task	8th Task	9th Task
AnyBoost	0.0040	0.0120	0.0437	0.0166	0.0292	0.0490	0.0078	0.0234	0.0232
	0.0003	0.0017	0.0034	0.0028	0.0076	0.0080	0.0027	0.0090	0.0066
MTGP	0.0009	0.0050	0.0374	0.0139	**0.0252**	0.0484	0.0069	0.0232	0.0230
	0.0005	0.0028	0.0035	0.0025	0.0073	0.0065	0.0034	0.0085	0.0087
Multi-boost	0.0010	0.0107	0.0421	0.0153	0.0276	0.0481	0.0065	0.0249	0.0259
	0.0363	0.0082	0.0088	0.0037	0.0021	0.0085	0.0041	0.0078	0.0078
MTRL	0.0008	0.0048	0.0353	0.0120	0.0267	**0.0340**	**0.0029**	0.0225	0.0223
	0.0007	0.0011	0.0035	0.0044	0.0014	0.0040	0.0019	0.0026	0.0046
MTBoost	**0.0003**	**0.0040**	**0.0324**	**0.0105**	**0.0252**	**0.0348**	0.0052	**0.0206**	**0.0213**
	0.0005	0.0034	0.0056	0.0042	0.0094	0.0075	0.0017	0.0060	0.0095

tasks. MTGP, another method which can learn the task covariance matrix from data, performs better than AnyBoost on some tasks but worse on other tasks. One possible reason is that MTGP usually uses low-rank approximation for the task covariance matrix to reduce the computational cost. This may affect the expressive power of the model and impair its performance. Moreover, MTBoost outperforms Multi-boost, MTGP and MTRL.

4.3 USPS Digit Classification

The USPS digit data set[5] contains 7,291 examples each of which is described by 255 features. There are nine classification tasks, each corresponding to the classification of two digits. The experimental settings are similar to those in the above subsections. The mean classification errors and the standard deviations of different methods over 10 trials are summarized in Table 5. Again, we find that MTBoost outperforms AnyBoost, Multi-boost and MTGP on almost every task. Moreover, MTBoost performs better than MTRL on most tasks and comparable with MTRL on the other tasks.

5 Conclusion

In this paper, we have proposed a multi-task boosting method based on learning and exploiting the pairwise relationships between tasks. The alternating optimization procedure in MTBoost has been shown to converge with theoretical guarantee. In the current setting, each task is a binary classification problem. One future direction is to extend this work to a general setting where each task can be either a binary or a multi-class classification problem. Besides, we mainly consider classification problems in this paper. In our future work, we will also investigate the extension of our method to regression problems. One possibility is to make use of the loss function defined in [28] for such problems.

[5] http://multitask.cs.berkeley.edu/

Acknowledgment

This research has been supported by General Research Fund 621310 from the Research Grants Council of Hong Kong.

References

1. Caruana, R.: Multitask learning. Machine Learning **28**(1) (1997) 41–75
2. Baxter, J.: A Bayesian/information theoretic model of learning to learn via multiple task sampling. Machine Learning **28**(1) (1997) 7–39
3. Thrun, S.: Is learning the n-th thing any easier than learning the first? In Touretzky, D.S., Mozer, M., Hasselmo, M.E., eds.: Advances in Neural Information Processing Systems 8, Denver, CO (1996) 640–646
4. Argyriou, A., Evgeniou, T., Pontil, M.: Multi-task feature learning. In Schölkopf, B., Platt, J.C., Hoffman, T., eds.: Advances in Neural Information Processing Systems 20, Vancouver, British Columbia, Canada (2007) 41–48
5. Argyriou, A., Evgeniou, T., Pontil, M.: Convex multi-task feature learning. Machine Learning **73**(3) (2008) 243–272
6. Evgeniou, T., Pontil, M.: Regularized multi-task learning. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA (2004) 109–117
7. Thrun, S., O'Sullivan, J.: Discovering structure in multiple learning tasks: The TC algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, Bari, Italy (1996) 489–497
8. Bakker, B., Heskes, T.: Task clustering and gating for bayesian multitask learning. Journal of Machine Learning Research **4** (2003) 83–99
9. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task learning for classification with Dirichlet process priors. Journal of Machine Learning Research **8** (2007) 35–63
10. Bonilla, E., Chai, K.M.A., Williams, C.: Multi-task Gaussian process prediction. In Platt, J., Koller, D., Singer, Y., Roweis, S., eds.: Advances in Neural Information Processing Systems 20, Vancouver, British Columbia, Canada (2008) 153–160
11. Zhang, Y., Yeung, D.Y.: Multi-task learning using generalized t process. In: Proceedings of the 13rd International Conference on Artificial Intelligence and Statistics, Chia Laguna Resort, Sardinia, Italy (2010) 964–971
12. Zhang, Y., Yeung, D.Y.: A convex formulation for learning task relationships in multi-task learning. In: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, Catalina Island, California (2010) 733–742
13. Zhang, Y., Yeung, D.Y., Xu, Q.: Probabilistic multi-task feature selection. In Lafferty, J., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A., eds.: Advances in Neural Information Processing Systems 23, Vancouver, British Columbia, Canada (2010) 2559–2567
14. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proceedings of the 13th International Conference on Machine Learning, Bari, Italy (1996) 148–156
15. Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods. The Annals of Statistics **26**(5) (1998) 1651–1686
16. Reyzin, L., Schapire, R.E.: How boosting the margin can also boost classifier complexity. In: Proceedings of the Twenty-Third International Conference on Machine Learning, Pittsburgh, Pennsylvania, USA (2006) 753–760
17. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. The Annals of Statistics **28**(2) (2000) 337–407

18. Friedman, J.: Greedy function approximation: A gradient boosting machine. The Annals of Statistics **29**(5) (2001) 1189–1232
19. Wang, X., Zhang, C., Zhang, Z.: Boosted multi-task learning for face verification with applications to web image and video search. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Miami, Florida, USA (2009) 142–149
20. Chapelle, O., Shivaswamy, P., Vadrevu, S., Weinberger, K., Zhang, Y., Tseng, B.: Multi-task learning for boosting with application to web search ranking. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA (2010) 1189–1198
21. Gupta, A.K., Nagar, D.K.: Matrix Variate Distributions. Chapman & Hall (2000)
22. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent. In Solla, S.A., Leen, T.K., Müller, K.R., eds.: Advances in Neural Information Processing Systems 12, Denver, Colorado, USA (1999) 512–518
23. Lange, K., Hunter, D.R., Yang, I.: Optimization transfer using surrogate objective functions. Journal of Computational and Graphical Statistics **9**(1) (2000) 1–59
24. Duchi, J., Singer, Y.: Boosting with structural sparsity. In: Proceedings of the 26th International Conference on Machine Learning, Montreal, Quebec, Canada (2009) 297–304
25. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: Proceedings of the 24th International Conference on Machine Learning, Corvalis, Oregon, USA (2007) 193–200
26. Pardoe, D., Stone, P.: Boosting for regression transfer. In: Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel (2010) 863–870
27. d'Alché-Buc, F., Grandvalet, Y., Ambroise, C.: Semi-supervised marginboost. In Dietterich, T.G., Becker, S., Ghahramani, Z., eds.: Advances in Neural Information Processing Systems 14, Vancouver, British Columbia, Canada (2001) 553–560
28. Zemel, R.S., Pitassi, T.: A gradient-based boosting algorithm for regression problems. In Leen, T.K., Dietterich, T.G., Tresp, V., eds.: Advances in Neural Information Processing Systems 13, Denver, CO, USA (2000) 696–702