Mach Learn (2011) 85:149–173
DOI 10.1007/s10994-010-5231-6

Boosted multi-task learning

Olivier Chapelle · Pannagadatta Shivaswamy · Srinivas Vadrevu · Kilian Weinberger · Ya Zhang · Belle Tseng

Received: 27 February 2010 / Revised: 27 September 2010 / Accepted: 25 November 2010 /
Published online: 24 December 2010

Abstract In this paper we propose a novel algorithm for multi-task learning with boosted decision trees. We learn several different learning tasks with a joint model, explicitly addressing their commonalities through shared parameters and their differences with task-specific ones. This enables implicit data sharing and regularization. Our algorithm is derived using the relationship between ℓ_1-regularization and boosting. We evaluate our learning method on web-search ranking data sets from several countries. Here, multi-task learning is particularly helpful as data sets from different countries vary largely in size because of the cost of editorial judgments. Further, the proposed method obtains state-of-the-art results on a publicly available multi-task dataset. Our experiments validate that learning various tasks jointly can lead to significant improvements in performance with surprising reliability.

Keywords Multi-task learning · Boosting · Decision trees · Web search · Ranking

Editors: Süreyya Özöğür-Akyüz, Devrim Ünay, and Alex Smola.

O. Chapelle (✉) · S. Vadrevu · B. Tseng
Yahoo! Labs, Sunnyvale, CA, USA
e-mail: chap@yahoo-inc.com

S. Vadrevu
e-mail: svadrevu@yahoo-inc.com

B. Tseng
e-mail: belle@yahoo-inc.com

P. Shivaswamy
Department of Computer Science, Cornell University, Ithaca, NY, USA
e-mail: pannaga@cs.cornell.edu

K. Weinberger
Washington University, Saint Louis, MO, USA
e-mail: kilian@wustl.edu

Y. Zhang
Shanghai Jiao Tong University, Shanghai, China
e-mail: ya_zhang@sjtu.edu.cn

1 Introduction

Multi-task learning algorithms (Caruana 1997) aim to improve the performance of several learning tasks through shared models. In this paper, we introduce a novel multi-task learning algorithm for gradient boosting. This is motivated by our interest in web search ranking, where gradient boosted decision trees are state-of-the-art methods (Li et al. 2008; Zheng et al. 2008).

Web search ranking is often treated as a supervised machine learning problem (Burges et al. 2005; Zheng et al. 2008): each query-document pair is represented by a high-dimensional feature vector and its label indicates the document's degree of relevance to the query. Like many other supervised learning problems, machine learned ranking requires a large number of labeled training examples, which are time consuming and expensive to obtain. This problem becomes more acute with specialization: most major search engines offer specialized rankings for different countries or regions as shown in Fig. 1. Queries like "football" issued in the UK generally should lead to different results than in the US. The problem of high editorial cost becomes even more prominent if one attempts to build many such specialized, country-specific ranking functions—as building each ranking function requires its own set of hand-labeled data. On the other hand, a large fraction of queries are region-insensitive. Thus, it seems worthwhile to treat the different markets as tasks that are not completely independent of one another as they share some commonalities, yet, differ enough that one cannot naïvely combine their training data sets.

There has been extensive research on multi-task learning in the past decade starting with the early works of Caruana (1997), Thrun (1996). Multi-task learning has been explored by various authors in different frameworks. This has lead to multi-task learning using kernel-methods (Argyriou et al. 2007, 2008; Evgeniou et al. 2006; Evgeniou and Pontil 2004), probabilistic approaches (Bakker and Heskes 2003; Yu et al. 2005; Xue et al. 2007), maximum-entropy discrimination (Jebara 2004), hashing (Weinberger et al. 2009) as well as theoretical work (Ben-David and Schuller 2003; Maurer 2006). However, despite the theoretical and empirical success of boosting, there are surprisingly few papers that consider multi-task learning in a boosting framework. In fact, we are not aware of any work other than (Wang et al. 2009) (which combines ideas from probabilistic methods and boosting) on multi-task learning in a boosting framework.

In this paper, we propose a novel algorithm to capture task specifics and commonalities simultaneously. Given data from T different tasks, the idea is to learn $T + 1$ models—one for each specific task and one global model that captures the commonalities amongst them. The algorithm is derived systematically based on the connection between boosting and ℓ_1 regularization (Rosset et al. 2004). We are not aware of any work that jointly models several tasks by explicitly learning both the commonalities and idiosyncrasies through gradient boosted regression.

Fig. 1 Search engines have specialized ranking functions for particular countries. Traditionally, these specialized search engines are trained independently of each other

Our contribution in this paper is three-fold:

1. We introduce a novel multi-task learning algorithm based on gradient boosted decision trees—the first of its kind.
2. We exploit the connections between boosting and ℓ_1 regularization to motivate the algorithm.
3. Driven by the success of gradient boosted decision trees on web-search, we perform a detailed evaluation on web-scale datasets.

The rest of the paper is organized as follows. After reviewing related work in Sect. 2, we formally introduce the multi-task learning problem in Sect. 3 and propose our approach in Sect. 4. The connection between boosting and ℓ_1 regularization serves as motivation for our derivations. The details of machine learning for ranking are given in Sect. 5. Then in Sects. 6 and 7, we evaluate our method on several real-world large scale web-search ranking data sets from different countries. Although we mainly focus on multi-task learning for ranking across different countries, we show in Sect. 8 that our multi-task algorithm can also be used for customizing ranking functions for different query types; and also experimental results on a public dataset for regression are reported in Sect. 9.

2 Related work

A common related line of research to multi-task learning is domain adaptation (DA). Here, one assumes a *source* domain with large training data and a *target* domain with very little training data. The test case is exclusively in the *target* domain. The main principle behind DA is to learn a model for the *source* and adapt it to the *target* domain. In the web-search example the source could be a well established country and the target a new country where the search engine is still relatively new. Gao et al. (2009) address this particular case with boosted decision trees through model interpolation and refinement. Although the authors are motivated by a very similar problem—adapting search engines to other countries—they choose a fundamentally different approach from us. While we train many functions (including a shared model) at the same time, they use an additive approach in which they first learn a base function and then train only *one* country-specific function.

Also related is the work on EM based multi-task boosting for face verification (Wang et al. 2009). Even though this work proposes a multi-task learning framework based on boosting, it is significantly different from our approach. With data from m tasks, it learns $k \leq m$ boosted classifiers. A multinomial variable indicates how well each classifier performs on each task. The proposed method uses an EM algorithm to maximize the log-likelihood (based on a probability model) where the M-step involves gradient boosting.

Another approach to multi-task learning is by assuming that the tasks share a common underlying representation. This viewpoint has lead to algorithms that learn a classifier, as well as a joint shared representation simultaneously (Argyriou et al. 2007; Chen et al. 2009). In a sense, multi-boost does this already; the mapping given by the weak learner is a common feature representation. A function is then learned on top of such shared representation.

3 Background

Notation and setup Assume that we are given learning tasks $t \in \{1, 2, \ldots, T\}$. Further, the data for these tasks, $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, is also given to us. Each task t is associated

with a set of indices I^t that denotes the data for this particular task. These index sets form a partition of $\{1,\ldots,n\}$ (*i.e.*, $I^t \cap I^s = \emptyset$ when $t \neq s$ and $\bigcup_{t=1}^{T} I^t = \{1,\ldots,n\}$). We also define $I^0 = \{1,\ldots,n\}$. At this point, we assume that all the tasks share the same feature space; we will later show how this assumption can be relaxed. Finally, we suppose that we are given a cost function C^t defined as a function of the predicted values for all points in I^t. For instance, in a regression setting, we might consider the squared loss:

$$C^t(\ldots,u_i,\ldots)_{i\in I^t} := \sum_{i\in I^t}(y_i-u_i)^2. \tag{1}$$

We also overload the definition of C^t to allow it to be defined as a function of the parameters of the function to be learned. For instance, in case of a linear class of functions,

$$C^t(w) := C^t(\ldots,\langle w,x_i\rangle,\ldots)_{i\in I^t}. \tag{2}$$

Previous work Previous work has mainly focused on Neural Networks (Caruana 1997; Collobert and Weston 2008) or Support Vector Machines (Evgeniou and Pontil 2004). This latter work inspired the algorithm presented in this paper. The authors adapt SVMs to multi-task learning by associating one classifier w_t specifically for each task t. In addition, there is a global classifier w^0 that captures what is common among all the tasks. The joint optimization problem is then to minimize the following cost:

$$\min_{w^0,w^1,\ldots,w^T} \sum_{t=0}^{T}\lambda_t\|w^t\|_2^2 + \sum_{t=1}^{T} C^t(w^0+w^t). \tag{3}$$

In Evgeniou and Pontil (2004), all the $\lambda_i,\ i\geq 1$ have the same value, but λ_0 can be different. Also, a classification task is considered: the labels are $y_i \in \{\pm 1\}$ and the loss function is

$$C^t(w) = \sum_{i\in I^t}\max(0,1-y_i\langle w,x_i\rangle). \tag{4}$$

Note that the relative value between λ_0 and the other λ_i controls the strength of the connection between the tasks. In the extreme case, if $\lambda_0 \to +\infty$, then $w_0 = 0$ and all tasks are decoupled; on the other hand, when $\lambda_i \to +\infty,\ \forall i \geq 1$, we obtain $w_i = 0$ and all the tasks share the same decision function with weights w^0.

Kernel-trick In practice, the formulation (3) suffers from the draw-back that it only allows linear decision boundaries. This can be very limiting especially for more complex real-world problems. A standard method to avoid this limitation is to apply the *kernel-trick* (Schölkopf and Smola 2002) and map the input vectors *indirectly* into a high dimensional feature space, $x_i \to \phi(x_i)$, where, with careful choice of ϕ, the data is often linearly separable. The kernel-trick is particularly powerful, because the mapping ϕ is implicitly chosen such that the inner-product between two vectors $\phi(x_i)^\top\phi(x_j)$ can be pre-computed very efficiently—even if the dimensionality of $\phi(x_i)$ is very high or infinite. The kernel-trick has been adapted to multi-task learning by Evgeniou and Pontil (2004), Caponnetto et al. (2008). Unfortunately, for many real-world problems, the quadratic time and space complexity on the number of input vectors is often prohibitive.

Hashing-trick In certain domains, such as text classification, it can be the case that the data is indeed linearly separable—even without the application of the kernel-trick. However, the input space $\mathcal{X}$ is often already so high dimensional, that the dense weight vectors w^t become too large for learning to be feasible—especially when the number of tasks T becomes very large. Recently Weinberger et al. (2009) applied the *hashing-trick* to a non-regularized variation of (3) and mapped the input data of all tasks into a single *lower* dimensional feature space. Similar to the kernel-trick, the high-dimensional representation is never computed explicitly and all learning happens in the compact representation.

4 Multi-boost

In this paper we focus on the case where the data is too large to apply the kernel-trick and not linearly separable, which is a key assumption for the hashing-trick. As already noted in the introduction, boosted decision trees are very well suited for our web search ranking problem and we now present our algorithm, *multi-boost* for multi-task learning with boosting.

4.1 Boosting-trick

Instead of mapping the input features into a high dimensional feature space with cleverly chosen kernel functions, we propose to use a set of non-linear functions $\mathcal{H} = \{h_1, \ldots, h_J\}$ to define $\phi : \mathcal{X} \to \mathbf{R}^J$ as $\phi(x_i) = [h_1(x_i), \ldots, h_J(x_i)]^\top$. Instead of assuming that we can compute inner-products efficiently (as in the kernel-trick), we assume that we are provided with an oracle $\mathcal{O}$ that solves the least-squared regression problem efficiently up to ϵ accuracy:

$$\mathcal{O}(\{(x_i, z_i)\}) \approx \operatorname*{argmin}_{h \in \mathcal{H}} \sum_i (h(x_i) - z_i)^2, \tag{5}$$

for some targets z_i. For the sake of the analysis we assume that $|\mathcal{H}| = J$ is finite, but in practice, we used regression trees and $\mathcal{H}$ is infinite.

Even though J may be very large, it is possible to learn linear combinations of functions in $\mathcal{H}$ using the so-called *boosting trick*. Viewing boosting as a coordinate descent optimization in a large space is of course not new and was first pointed out in (Mason et al. 2000). The contribution of this paper is the adaptation of this insight to multi-task learning.

Let us apply the boosting-trick to the optimization problem (3). For disambiguation purposes, we denote the weight vector for task t in $\mathbf{R}^J$ as β^t. As J can be very large, we can only store vectors $\beta \in \mathbf{R}^J$ if they are extremely sparse. For this reason, we change the regularization in (3) from an ℓ_2-norm to an ℓ_1-norm. We can state our modified multi-task learning formulation as

$$\begin{aligned} &\min_{\beta^0, \beta^1, \ldots, \beta^T} \sum_{t=1}^{T} C^t(\beta^0 + \beta^t) \\ &\text{under constraint } \sum_{t=0}^{T} \lambda_t \|\beta^t\|_1 \leq \mu, \end{aligned} \tag{6}$$

where, as in (2), $C^t(\beta)$ is defined as $C^t(\ldots, \langle \beta, \phi(x_i) \rangle, \ldots)_{i \in I_t}$. A minor technical difference from (3) is that the regularizer is introduced as a constraint. We do not make any explicit assumptions on the loss functions $C^t(\cdot)$, except that it needs to be differentiable.

Similar to the use of the kernel-trick, our new feature representation $\phi(x_i)$ forces us to deal with the problem that in most cases the feature space $\mathbf{R}^J$ will be extremely high dimensional. For example, for our experiments in the result section, we set $\mathcal{H}$ to be the set of regression trees (Breiman et al. 1984)—here $|\mathcal{H}|$ is infinite and $\phi(x_i)$ cannot be explicitly computed. To the rescue comes the fact that we will never actually have to compute $\phi(x_i)$ explicitly and that the weight vector β^t can be made sufficiently sparse with the ℓ_1-regularization in (6).

4.2 Boosting and ℓ_1 regularization

In this section we will derive an algorithm to solve (6) efficiently. In particular we will follow previous literature by Rosset et al. (2004) and ensure that our solver is in fact an instance of gradient boosting (Mason et al. 2000). To simplify notation, let us first transform the multi-task optimization (6) into a traditional single-task optimization problem by stacking all parameters into a single vector $\beta \in \mathbf{R}^{J(T+1)}$, defined as:

$$\beta := \begin{pmatrix} \beta^0 \\ \vdots \\ \beta^T \end{pmatrix} \quad \text{and} \quad C(\beta) := \sum_{t=1}^{T} C^t(\beta^0 + \beta^t). \tag{7}$$

This reduces (6) to the following optimization problem:

$$\min_{\|\beta\|_\lambda \leq \mu} C(\beta), \tag{8}$$

where we define the norm

$$\|\beta\|_\lambda = \sum_{t=0}^{T} \lambda_t \|\beta^t\|_1.$$

The goal in this section is to find an algorithm that solves (8) without ever computing any vector $\phi(x_i)$ explicitly.

ϵ-boosting As a first step, let us define a simple iterative algorithm to solve (8) that Rosset et al. (2004) refers to as *ϵ-boosting*. Intuitively, the idea is to follow the regularization path as μ is slowly increased from 0 to the desired value in tiny $\epsilon > 0$ increments. This is possible under the assumption that the optimal vector β in (8) is a monotonic function of μ componentwise.[1] At each iteration, the vector β is updated only incrementally by an additive factor of $\Delta\beta$, with $\|\Delta\beta\|_\lambda \leq \epsilon$. More precisely, $\Delta\beta$ is found through the following optimization problem:

$$\min_{\Delta\beta} C(\beta + \Delta\beta) \quad \text{s.t.} \quad \|\Delta\beta\|_\lambda \leq \epsilon. \tag{9}$$

Following Rosset et al. (2004), it can be shown that, under the monotonicity assumption stated above, solving (9) for the right number of iterations does in fact solve (7). Therefore, ϵ-boosting satisfies the ℓ_1 regularization constraint from (6) *implicitly*. Because the ℓ_1-norm

[1] In practice this assumption will only be partly satisfied. ϵ-boosting should thus be considered as a way to obtain an *approximate* solution of the ℓ_1 regularized optimization problem.

of the vector β increases by at most ϵ during each iteration, the regularization is not controlled by the upper bound μ but instead by the number of iterations S for which the vector β is updated. In particular, after S iterations, the ℓ_1 norm of the solution will be bounded by $S\epsilon$.

Multi-task ϵ-boosting As we are only moving in very tiny ϵ steps, it is fair to approximate $C(\cdot)$ with the first-order Taylor expansion. By "unstacking" the representation from (7), this leads to

$$C(\beta + \Delta\beta) \approx C(\beta) + \sum_{t=0}^{T} \langle \Delta\beta^t, g^t \rangle, \tag{10}$$

with $g_j^t := \frac{\partial C}{\partial \beta_j^t}$.

Let us define the outputs at the training points as

$$u_i = \langle \beta^0, \phi(x_i) \rangle + \langle \beta^t, \phi(x_i) \rangle \quad \text{for } i \in I^t,\ t > 0. \tag{11}$$

Using the chain-rule,

$$g_j^t = \sum_{i=1}^{n} \frac{\partial C(u)}{\partial u_i} \frac{\partial u_i}{\partial \beta_j^t}.$$

On the other hand,

$$\frac{\partial u_i}{\partial \beta_j^t} = \begin{cases} h_j(x_i) & \text{if } i \in I_t, \\ 0 & \text{otherwise.} \end{cases}$$

Combining the above equations, we finally obtain:

$$g_j^t = \sum_{i \in I^t} \frac{\partial C(u)}{\partial u_i} h_j(x_i). \tag{12}$$

We can now rewrite our optimization problem (9) with the linear approximation (10) as:

$$\min_{\Delta\beta^t} \sum_{t=0}^{T} \langle \Delta\beta^t, g^t \rangle \quad \text{s.t.} \quad \sum_{t=0}^{T} \lambda_t \|\Delta\beta^t\|_1 \le \epsilon. \tag{13}$$

If we make the additional assumption that the class of functions $\mathcal{H}$ is closed under negation ($h \in \mathcal{H} \Rightarrow -h \in \mathcal{H}$) then it is relatively easy to show (see appendix) that the solution of (13) is given by:

$$\Delta\beta_j^t = \begin{cases} \frac{\epsilon}{\lambda_t}, & \text{if } (t, j) = \operatorname{argmin}_{(t,j)} \frac{g_j^t}{\lambda_t}, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Intuitively, (14) finds the direction with steepest descent, across all tasks and all functions in $\mathcal{H}$, and takes a step in that direction.

It remains to show that we can compute the single non-zero index of (14) efficiently with the help of the oracle (5). Assuming that the weighted functions $h \in \mathcal{H}$ are normalized over

the input,[2] that is, $\sum_{i\in I^t} h(x_i)^2 = 1$, we can express (5) with $z_i = -\frac{\partial C(u)}{\partial u_i}$ as

$$\begin{aligned}\operatorname*{argmin}_j \sum_{i\in I^t}(h_j(x_i) - z_i)^2 &= \operatorname*{argmin}_j \sum_{i\in I^t} -h_j(x_i)z_i \\ &= \operatorname*{argmin}_j \; g_j^t \\ &:= \hat{\jmath}(t).\end{aligned}$$

The optimal task-feature pair (t, j) from (14) is thus $(\hat{t}, \hat{\jmath}(\hat{t}))$ with

$$\hat{t} = \operatorname*{argmax}_t \; \frac{1}{\lambda_t} \sum_{i\in I^t} h_{\hat{\jmath}(t)}(x_i) z_i. \tag{15}$$

The parametrization in terms of β is just conceptual and in practice we update the function $F^t(\cdot) := \langle \beta^t, \phi(\cdot)\rangle$ instead of β:

$$F^{\hat{t}}(\cdot) \leftarrow F^{\hat{t}}(\cdot) + \epsilon h_{\hat{\jmath}(\hat{t})}(\cdot). \tag{16}$$

The length of the step in (16) is ϵ instead of $\epsilon/\lambda_{\hat{t}}$ as indicated in (14). However since ϵ is supposed to be infinitesimally small, both are equivalent. The algorithm *multi-boost* with the update-rule (16) is summarized in algorithm 1 and is illustrated in Fig. 2.

4.3 Generalizations

For the sake of simplicity, we have tried to keep the above derivation as simple as possible, but there are in fact several extensions that we have implemented:

Different feature sets The training points from different tasks may have different features, and in fact, that is the case in our web search ranking application. To address this issue, we introduce, for each task t, a set of functions $\mathcal{H}^t$ defined over the features for that task. $\mathcal{H}^0$ is

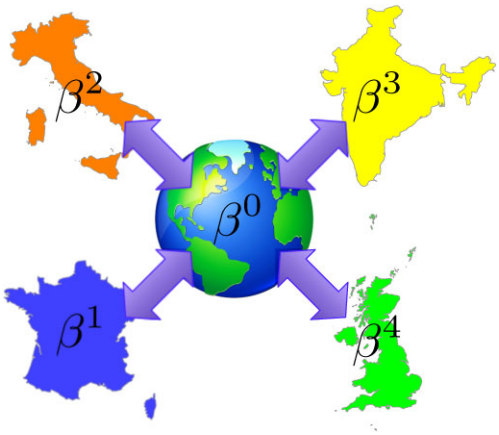

Fig. 2 A layout of four ranking tasks that are learned jointly. The four countries symbolize the different ranking functions that need to be learned, where $\beta^1, \ldots, \beta^4$ are the parameter vectors that store the specifics of each individual task. The various tasks interact through the joint model, symbolized as a globe with parameter vector β^0. More complicated (e.g. pairwise) interactions could be allowed through additional cross-task links

[2] In general this cannot hold for all t simultaneously—an easy fix is to make h dependent on t, as we will show later on.

the set of functions defined over the intersection of the various feature sets. That does not change the algorithm fundamentally; the main difference is that now u_i in (11) is defined as $\langle \beta^0, \phi^0(x_i) \rangle + \langle \beta^t, \phi^t(x_i) \rangle$, where ϕ^0 and ϕ^t are defined with respect to the functions of $\mathcal{H}^0$ and $\mathcal{H}^t$ respectively.

Second order information Instead of performing an ϵ-gradient step, we follow the framework of Zheng et al. (2008) and perform an approximate Newton step at each iteration. At the core of this approach is the computation of the second derivatives of the loss, $t_i = \frac{\partial^2 C}{\partial u_i^2}$, and the use of a *weighted* least square algorithm as an oracle with weights t_i. More details can be found later in Sect. 5 and Algorithm 2.

Weights We introduce a weight c^t for each task t such that the new global objective function is $C := \sum c^t C^t$. We experiment with two choices for c^t: $c^t = 1$ and $c^t = \frac{1}{|I^t|}$.

Algorithm 1 Multi-boost (S iterations)

$F^t = 0 \quad \forall 0 \leq t \leq T$
for $s \leftarrow 1$ to S **do**
  $z_i = -\frac{\partial C(u)}{\partial u_i} \quad \forall 1 \leq i \leq n$
  $\hat{h}^t \leftarrow \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_{i \in I^t} (h(x_i) - z_i)^2, \quad 0 \leq t \leq T.$
  $\hat{t} \leftarrow \underset{t}{\operatorname{argmax}} \frac{1}{\lambda_t} \sum_{i \in I^t} \hat{h}^t(x_i) z_i.$
  $F^{\hat{t}} \leftarrow F^{\hat{t}} + \epsilon \hat{h}^{\hat{t}}$
  $u_i \leftarrow u_i + \epsilon \hat{h}^{\hat{t}}(x_i) \quad \forall i \in I^t$
end for
Predict a new example x of task t as $F^0(x) + F^t(x)$

4.4 Extensions

We finish with two extensions that we have not yet tested, but are rather straightforward to implement:

Multiple interactions So far we assumed that the T tasks only interact through a single global model F^0 (as it is illustrated in Fig. 2). It is possible to allow more interactions by adding additional sets $I^{T+1}, \ldots, I^{T+k}$. For example I^{T+1} could explicitly model the pairwise interaction between two tasks t_1, t_2 by setting $I^{T+1} = I^{t_1} \cup I^{t_2}$. The final prediction for an example x_i becomes $\sum_{t \in \text{Tasks}(x)} F^t(x)$, where $\text{Tasks}(x)$ contains all the real and artificial tasks to which x belongs.

AdaBoost style algorithm In binary classification problems, it is possible to motivate an AdaBoost style algorithm with multiple tasks. In this case: $y_i \in \{\pm 1\}$ and $h : \mathcal{X} \to \{\pm 1\}$. Based on the stage-wise greedy minimization of interpretation of AdaBoost, it is straightforward to derive an update rule to minimize $\sum_{t=1}^{T} C^t((\beta^0 + \Delta\beta^0) + (\beta^t + \Delta\beta^t))$ in $\Delta\beta$, where, $\Delta\beta$ now has all but one of the elements set to zero, C^t is the exponential loss. The resulting algorithm either updates the weights for all the examples (if it is a global step) or for all the examples of a specific task (if it is a local step).

5 Machine learned ranking

Document retrieval has traditionally been based on a manually designed ranking function (e.g., BM25). However, web page ranking is now considered as a supervised learning problem, and several machine learning algorithms have already been applied to it. One of the earliest publications in the context of web page ranking is Burges et al. (2005), where a neural network is trained on pairwise preferences. Pairwise learning is now one of the most popular techniques: it can be used directly with linear functions (Cao et al. 2006) or it can be combined within the boosted decision tree framework (Zheng et al. 2008). Recent work has addressed the possibility of directly optimizing ranking measures such as MAP (Yue et al. 2007) or NDCG (Taylor et al. 2008; Chapelle and Wu 2010). Finally, it is noteworthy that a simple regression on regression labels (Cossock and Zhang 2006) turns out to be competitive compared to more advanced techniques (Li et al. 2008).

Functional gradient boosting The functional gradient boosting framework introduced by Friedman (2001) has already been discussed in Sect. 4. We now present more details about a second order extension introduced in Zheng et al. (2008). It is indeed this extension that we use as our underlying boosting algorithm. Let $\mathcal{R}(f)$ be an objective function which depends on the value of the function evaluated at the training points:

$$\mathcal{R}(f(x_1), \ldots, f(x_n)).$$

In the case of least-squares regression—as in GBDT (Friedman 2001)—this objective function is simply:

$$\sum_{i=1}^{n} (f(x_i) - y_i)^2.$$

But in the case of ranking, it can be beneficial to define the objective function as a sum of pairwise losses over preference pairs, as in RankSVM (Herbrich et al. 2000) and GBRank (Zheng et al. 2008):

$$\sum_{(i,j)\in\mathcal{P}} \max(0, 1 - (f(x_i) - f(x_j)))^2,$$

where $\mathcal{P}$ is a set of preference pairs: $(i, j) \in \mathcal{P}$ mean that the i-th document is to be preferred to the j-th one. We thus refer in the experimental section to *GBDT* and *GBRank* as our learning methods for ranking.

Given a set $\mathcal{H}$ of base functions such as decision trees, the goal is to optimize $\mathcal{R}$ over the linear span of $\mathcal{H}$. In other words, we want to find a solution of the form:

$$f = \sum_i \beta_i h_i, \quad h_i \in \mathcal{H}.$$

The functional gradient boosting algorithm from Zheng et al. (2008) and described in Algorithm 2 is one way to do so and the one we will use in the rest of this paper. Compared to the original algorithm of Friedman (2001), it uses second order information to better approximate the objective function. More precisely, it uses the following quadratic approximation:

$$\mathcal{R}(f + h) \approx \mathcal{R}(f) + \sum_{i=1}^{n} \underbrace{\frac{\partial \mathcal{R}(f)}{\partial f(x_i)}}_{:=g_i} h(x_i) + \frac{1}{2} \sum_{i=1}^{n} \underbrace{\frac{\partial^2 \mathcal{R}(f)}{\partial f(x_i)^2}}_{:=w_i} h(x_i)^2. \tag{17}$$

Algorithm 2 Generic gradient boosting algorithm with second order approximation

$f \leftarrow 0$
repeat
  $g_j \leftarrow \frac{\partial \mathcal{R}(f)}{\partial f(x_j)}$. Functional gradient
  $w_j \leftarrow \frac{\partial^2 \mathcal{R}(f)}{\partial f(x_j)^2}$. Functional curvature
  $t_j \leftarrow -g_j/w_j$. Target value
  $\hat{\imath} \leftarrow \arg\min_i \sum_{j=1}^{n} w_i (h_i(x_j) - t_j)^2$. Steepest component
  $\hat{\rho} \leftarrow \arg\min \mathcal{R}(f + \rho h_{\hat{\imath}})$. Line search
  $f \leftarrow f + \eta \hat{\rho} h_{\hat{\imath}}$. "Shrinkage" when $\eta < 1$.
until Max iterations reached.

The right hand side of (17) can be written as:

$$\frac{1}{2} \sum_{i=1}^{n} w_i \left(h(x_i) + \frac{g_i}{w_i} \right)^2 + C,$$

where C is a constant independent of h. The quadratic approximation of $\mathcal{R}$ can thus be minimized by solving a weighted least squares problem where the targets are given by $t_i := -g_i/w_i$.

Features For each query-document pair a set of features is extracted to form a feature vector which typically consists of three parts:

Query-feature vector, comprising features depending only on the query q and have constant values across all the documents d in the document set, for example, the number of terms in the query, whether or not the query is a person name, etc.

Document-feature vector, comprising features depending only on the document d and have constant values across all the queries q in the query set, for example, the number of inbound links pointing to the document, the amount of anchor-texts in bytes for the document, and the language identity of the document, etc.

Query-document feature vector, comprising features which depend on the relation of the query q with respect to the document d, for example, the number of times each term in the query q appears in the document d, the number of times each term in the query q appears in the anchor-texts of the document d, etc.

The total number of features we use for our experiments is of the order of 500.

Data collection The data sets used in our experiments include large-scale web ranking training sets for various countries. All the data sets contain randomly sampled queries from the search engine query logs. For each query, a set of URLs is sampled from the results retrieved by several search engines. Finally, for each query-url pair, an editorial grade containing 0–4 is obtained that describes the relevance of the url to the query. The size in terms of number of queries and the number of query-url pairs in the training, test and validation sets for each country is shown in Table 1. The country names are anonymized for confidentiality purposes. Note that there are some empty cells for some countries which indicate that the test and validation sets are not available for them.

Table 1 Details of the subset of data used in experiments. The countries have been sorted in increasing size of the number of training queries

Country	Examples			Queries		
	Train	Validation	Test	Train	Validation	Test
A	72k	7k	11k	3486	477	600
B	64k	10k	–	4286	563	–
C	74k	4k	11k	5992	298	600
D	108k	12k	14k	7027	383	600
E	162k	14k	11k	7204	586	600
F	74k	11k	11k	7295	486	600
G	57k	5k	15k	7356	238	600
H	137k	11k	12k	7644	807	600
I	95k	12k	12k	8153	835	600
J	166k	12k	11k	11145	586	600
K	62k	10k	20k	11301	548	600
L	307k	–	–	12850	–	–
M	474k	–	–	15666	–	–
N	194k	16k	12k	18331	541	600
O	401k	–	–	33680	–	–

Evaluation The performance of various ranking models is measured on the test set using Discounted Cumulative Gain (Jarvelin and Kekalainen 2002), which is a popular measure for multi-level relevance judgments. In its basic form it has a logarithmic position discount: the benefit of seeing a relevant document at position i is $1/\log_2(i+1)$. Following (Burges et al. 2005), it became usual to assign exponentially high weight 2^{l_i} to highly rated documents where l_i is the grade of the i-th document going for instance from 0 (irrelevant) to 4 (perfect result). Thus the DCG for a ranking function r of a query having m associated documents is defined as:

$$\mathrm{DCG} := \sum_{i=1}^{m} \frac{2^{l_i} - 1}{\log_2(1 + r(i))},$$

where $r(i)$ is the position (or rank) of the i-th document in the ranking. We only consider the top 5 positions in the ranking and refer to the DCG of these 5 documents as DCG-5.

Model selection In most of the experiments, the parameters of the algorithms are tuned using a validation set. But for some of them—that we will point out—some parameters are set to default values which in general give good performances. These values are 20 for the number of nodes per tree, 1200 for the number of trees and 0.05 for the *shrinkage* rate (Friedman 2001).

6 Preliminary experiments

The main experimental results—multi-task learning applied to country specific web search ranking—have been divided into two sections. In this section we present preliminary results

on a small feature set containing 11 most important features such as a static rank for the page on the web graph, a text match feature and the output of a spam classifier. For the large-scale experiments in Sect. 7, we present the results with the complete feature set containing more than 500 features.

We did not use in this section the validation and test sets described in Table 1. Instead we split the given training set into a smaller training set, a validation set and a test set. The proportions of that split are, in average over all countries, 70%, 15% and 15% for respectively the training, validation and test sets. The reason for this construction will become clearer in the next section; in particular, the test sets from Table 1 were constructed by judging the documents that our candidates functions retrieved and cannot thus be used for experimentation but only as a final test.

We initially discuss the correlation between train MSE and test DCG. Later, we compare several baseline ranking models and discuss the effect of sample weighting.

For each experiment, we calculated the DCG on both validation set and the test set after every iteration of boosting. All parameters—the number of iterations, number of nodes in regression trees and the step size ϵ—where selected to maximize the DCG on the validation set and we report the corresponding test DCG.

Train MSE and test DCG We show how the train MSE and test DCG change for a typical run of the experiment in Fig. 3. The training loss always decreases with more iterations. The test DCG improves in the beginning and but the model starts overfitting at some point, and the DCG slightly deteriorates after that. Thus it is important to have a validation set to pick the right number of iterations as we have done. In the rest of the experiments in this section, we tuned the model parameters with the validation set and report the improvements over the test.

Baseline experiments We first did a smaller experiment on six countries. The aim in this experiment was to compare with the following baseline methods:

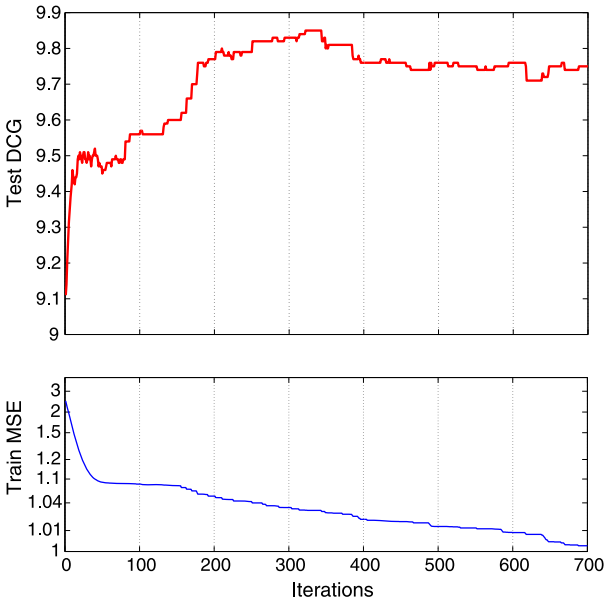

Fig. 3 Train MSE and test DCG as a function of iterations

Table 2 Percentage change in DCG over *independent* ranking models for various baseline ranking models

Country	*Weighted*	*Unweighted*	*Pooling*	*Cold-start*
A	0.561	1.444	−0.320	−0.282
C	1.135	1.295	0.972	1.252
D	−0.043	−0.233	−1.096	−2.378
E	0.222	0.342	−2.873	−3.624
M	−2.385	−0.029	−1.724	−6.376
N	−0.036	0.705	−1.160	−3.123

Independent: Each country trained on its own data;
Cold-start: Model trained using all the countries other than the local itself. The aim of this baseline was to see how much other countries could help a given country;
Pooling: All the countries are used in training. We ensured that the total weight on the local country was equal to that of all the other countries put together.

Table 2 summarizes the results relative to the *independent* baseline. The two heuristic schemes—*cold-start* and *pooling*—did not show any improvement overall (in fact, most DCG values were lower). Hence in all of the experiments that follow, we used the *independent* ranking model as the baseline and show that our multi-task learning algorithm can improve over the *independent* models as well. As described in Sect. 4.3, we can provide a weight for each data set in the multi-task learning scenario that we proposed. In this table the *unweighted* scheme refers to setting the weight as 1 for each example and *weighted* refers to weighting each data set by the reciprocal number of samples in the data set. Thus *weighted* gives equal weight to each data set, while *unweighted* has no weight on each sample, so effectively larger data sets have higher weight in the *unweighted* setting. The results indicate that the average performance of the *unweighted* scheme seems better than the *weighted* one. Note that a relative improvement of 1% is considered to be substantial in the web search ranking domain.[3]

Steps taken by the two weighting schemes Typical behavior of the steps taken with the two weighting schemes are shown in Fig. 4. In both the schemes, initially, a number of global steps are taken. Since global steps minimize the objective function for every country, it is attractive initially. However, once the commonality among the tasks has been captured by the global steps, they are no longer very attractive. The algorithm takes many local steps from that point onwards. Furthermore, with the *unweighted* scheme, the countries with significantly more data dominate the objective. Thus, the multi-task algorithm takes significantly more steps in such countries. On the other hand, in the *weighted* scheme, smaller countries are relatively easier to fit than bigger ones and a lot of steps are taken in these countries. Although we presented two extreme weighting schemes, other weighting schemes with specific weights to each country are possible.

Finding appropriate groups of countries Finding an appropriate grouping of countries that is beneficial to each country so that the tasks in that group can help each other is a nontrivial task. Since we wanted to find out the best group of countries that is most beneficial to each country, we searched all possible combinations of countries. Specifically we explored

[3]What we mean here is that improvements reported in web search ranking papers are typically of the order of 1%; see for instance Fig. 1 of Zheng et al. (2008).

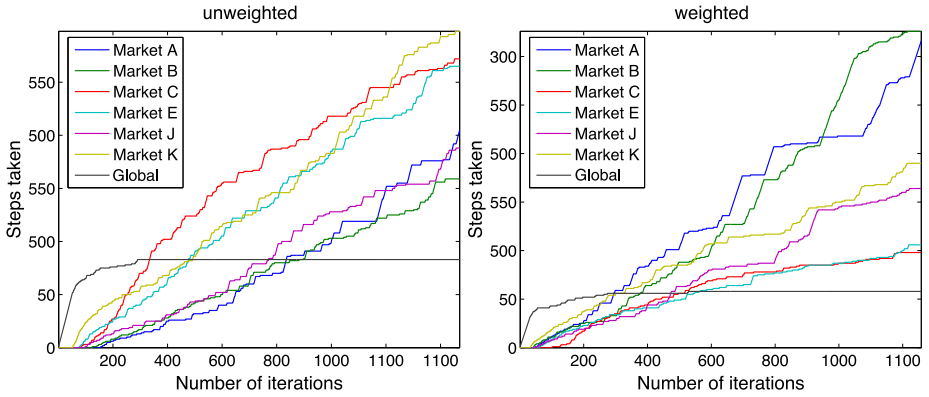

Fig. 4 (Color online) Steps taken by the multi-task algorithm with the two weighting schemes. The country labels A-K are sorted by increasing data set sizes. The *unweighted* (*left*) version takes more steps for the larger data sets, whereas the *weighted* (*right*) variation focusses more on countries with less data

Table 3 Percentage improvement over *independent* for the best countries found on the validation set

Country	% gain	Best countries
A	+4.21	C D F H L M N
B	+2.06	N
C	+1.70	A M
D	+2.95	C H L N
E	+0.35	B C F L O
F	+1.43	A B E H L N
H	+1.11	A B D E F L
L	+0.57	A B C E F M
M	+0.45	A C N
N	+1.00	A F L
O	+0.61	A F

2^{11} combinations of possible groupings for eleven countries and found the best group of countries that helps a given country based on the validation data set. Then we tested this best model on the test set to observe its performance. Since this experiment involves a very large number of combinations, we have fixed to some default values the learning parameters of the gradient boosted decision tree algorithm (number of nodes, shrinkage and number of trees). Table 3 shows the experimental results for this task. Each row shows the DCG-5 gain of the best grouped multi-task ranking model over the *independent* ranking model for each country. We can see that the multi-task ranking model improves the performance in every single country over the country-specific ranking model.

7 Large scale experiments

In this section we present the experimental results by testing the multi-task algorithms on the large scale web search ranking data sets with complete feature sets. We illustrate that our methods help to customize the ranking models on a number of country-specific data

sets. For Sects. 7.1, 7.2 and 7.3, we fixed the parameters such as number of trees, number of nodes per tree and compare *multi-task* ranking models with *independent* models on the validation set. In Sects. 7.4 and 7.5, we did complete model selection on the validation set and report the results on the test set.

Note that we have two different experimental settings in this paper. Most of the experiments are in a *reranking* setting, where a fixed set of documents with relevance grades are exposed to the ranking models for each query. This is the traditional setting used in almost all of the learning to rank papers. On the other hand, in Sects. 7.4 and 7.5, we present results based on *web-scale* experimental setting, where all the documents[4] in the web index are exposed to the ranking models. To our knowledge, we are the first to provide results in such a setting. It also serves as a further validation to the results obtained with the *reranking* experimental setting.

7.1 Effect of the loss function

As described in Sect. 4 our method can be applied to any loss function and in particular it can be combined with pairwise or listwise ranking models. In this section, we compare the results of pointwise and pairwise ranking schemes, which we refer to as *Multi-GBDT* and *Multi-GBRank* methods, to the *independent* ranking models in each country. Table 4 shows the results with the two learning algorithms in both the weighting schemes discussed in Sect. 6. It can be seen that in both pointwise and pairwise ranking schemes, *multi-task* ranking models have better average performance over the *independent* models.

7.2 Grouping related tasks

An important aspect of the multi-task learning algorithms that we proposed is that if we can group the tasks so that they are related and benefit each other, we can boost the performance

Table 4 DCG-5 gains with *Multi-GBDT* and *Multi-GBRank* learning algorithms in two different weighting settings. The gains are over *independent-GBDT* and *independent-GBRank* respectively

Country	Multi-GBDT		Multi-GBRank	
	Unweighted	Weighted	Unweighted	Weighted
A	+1.53	+0.72	+0.69	+0.75
B	+1.81	+1.58	+2.22	+1.64
C	+0.92	+0.52	+0.92	+0.01
D	+4.14	+3.62	+1.77	+1.84
E	−1.37	−1.45	−0.18	−0.91
F	+0.57	+1.80	+1.67	+2.22
G	+4.34	+4.68	+1.74	+0.75
H	+0.34	+0.96	+0.52	+0.85
I	−0.50	−0.80	−0.07	+0.32
J	+0.10	−0.69	+0.74	−0.64
K	+2.37	+2.38	+3.40	+2.01
N	+0.53	−1.23	+0.51	−0.92
Mean	+1.23	+1.01	+1.16	+0.66

[4]To be precise, it would be infeasible to score all the documents in the index and only the potentially relevant documents—as determined by a basic ranking function—are scored.

Table 5 Improvement of multi-task models with various groupings of countries over *independent* ranking models. Each column in the table indicates a group that includes a subset of countries and each row in the table corresponds to a single country. The numbers in the cell are filled only when the corresponding country is part of the corresponding group. The symbol † indicates that this country was included for training but has not been tested

Country	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7	Group 8
A	+0.33	+1.53				+0.46		
B	+1.87	+1.81		+1.60				
C	+1.81	+0.92				+0.64		
D	+2.90	+4.14		+2.29				
E	−1.02		−1.37					−1.24
F	+1.82	+0.57		−0.20				
G	+1.51		+4.34				+3.59	
H	+0.00	+0.34			+0.08			
I	−0.85	−0.50			−0.97			
J	+0.76		+0.10					+0.38
K			+2.37				+1.98	
L			†		†	†		
N			+0.53			−0.09		
O		†						†

of the individual tasks. To demonstrate the benefits of grouping the related tasks, we grouped the related countries into several groups and present results in Table 5. For each country, DCG-5 gain compared to the *independent* ranking model is shown for all the groups in which it was involved. The underlying learning algorithm in this experiment was GBDT and we used the *unweighted* scheme of our multi-task learning algorithm.

We organized the countries into eight groups based on continents and language of the countries and we anonymized the group names. Each of these groups involve a set of countries which are indicated in the columns of Table 5. It can be noted that different groups are beneficial to each country and finding an appropriate grouping that is beneficial to a specific country is challenging. The negative numbers for some countries indicate that none of the groups we selected were improving that particular country and the *independent* ranking model is still better than various groupings we tried.

7.3 Comparison over adaption models

In this section we present comparison results of our *multi-boost* method with the domain adaptation method described in Gao et al. (2009). In this domain adaptation setting, the key idea is to utilize the *source* data to help and adapt to the *target* domain. We chose the *source* data as the simple combination of data from all of the countries and varied the *target* domain. In addition, the adaptation method also requires an additional parameter in terms of the number of trees that are added at the end of the base model to train with the target domain data. We fixed the number of base model trees to be 1000 and the number of additive trees as 200. Figure 5 shows the DCG-5 gains of our method over the adaptation method in Gao et al. (2009). The multi-task method outperforms the adaptation method in most of the countries, while the adaptation method is better in a couple of countries. Since the multi-task method allows the domains to learn from each other, it fosters better interaction among the

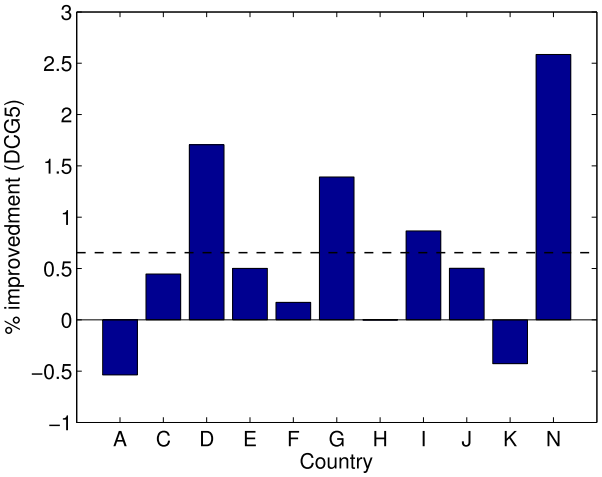

Fig. 5 DCG-5 gains of multi-task models over the adaption models. The *dashed line* represents the mean gain, 0.65%

domains than the adaptation method. Also a key difference with the adaption method is that multi-task automatically decides the number of steps taken towards each of the domains by choosing the domain that minimizes the objective function at each step. Moreover, since the steps are interleaved among the domains, the target data is introduced earlier to the model than the adaptation method.

7.4 Web scale experiments

The experimental results with *web-scale* experimental setting are shown in Table 6. To perform this test, several multi-task models were first trained with various parameters including the grouping and weighting as some of the parameters in addition to the standard GBDT parameters such as number of trees, number of nodes per tree and the shrinkage rate (Friedman 2001). Of these models, we picked the model with the highest DCG-5 on the validation set as the $\text{Best}_{\text{valid}}$ model. We also selected the top three models on the validation set as well as the *independent* model to be evaluated in the *web-scale* experimental setting. This means that all the top documents retrieved by these models were sent for editorial judgements. Of these three selected models, the one achieving the highest DCG-5 on the test set is denoted $\text{Best}_{\text{test}}$. Table 6 shows the improvements with both $\text{Best}_{\text{valid}}$ and $\text{Best}_{\text{test}}$ models.

The results indicate that the small tasks have much to benefit from the other big tasks where the training data size is large. It can also be noticed that the difference between reranking and web scale results is also dependent on the size of the validation or reranking data set. When the size of the validation set is large, there is more confidence on the results from the reranking results.

7.5 Experiments with global models

As discussed in Sect. 4, a byproduct of our multi-task learning algorithm with multiple countries is a *global* model, F^0 that is learned with data from all the countries. This global model can be utilized to deploy to the countries where there is no editorial data available at all, which could serve as a good generic model. Table 7 shows the results of the global models in the same *web-scale* setting as in the previous section. For each country we present the DCG-5 gains of the multi-task global models over the baseline ranking model that is

Table 6 Web scale results obtained by judging all the urls retrieved by the top 3 multi-task models as well as the *independent* model. $\text{Best}_{\text{valid}}$ refers to DCG-5 gain with the best model on the validation set while $\text{Best}_{\text{test}}$ refers to the highest DCG-5 gain on the test set

Country	$\text{Best}_{\text{valid}}$	$\text{Best}_{\text{test}}$
A	+2.99	+3.02
C	+2.31	+3.73
D	−1.03	+0.27
E	−0.02	+0.07
F	+2.12	+3.27
G	+4.80	+4.80
H	+3.27	+3.27
I	+0.10	+1.38
J	+4.04	+4.20
K	+6.85	+9.39
N	+2.11	+2.11

Table 7 DCG-5 gains of global models trained with multi-task approach compared with simple data combination from of all countries

Country	Improvements with F^0
A	+2.94
C	−0.20
D	−0.17
E	−0.33
F	+0.83
G	+0.49
H	+1.18
I	+0.73
J	+4.83
K	+0.85
N	−1.48
Mean	+0.88

trained with a simple combination of data from all countries. A key difference between these two models is that the multi-task global model primarily learns the commonalities in the countries while simple data combination model could learn both commonalities and the country specific idiosyncrasies. While these country specific idiosyncrasies are helpful for the that specific country, it might actually hurt other countries. Although the global model does not perform well in a few countries, the average performance of the multi-task global ranking model is better than the simple data combination model.

8 Query dependent ranking

Most of the learning to rank approaches, including the ones presented in this paper, attempt at learning a relevance function which can then be applied to any query. But there are different types of queries and it might be beneficial to have a different ranking function for each query class. For instance, Kang and Kim (2003) proposed to used different ranking functions for navigational and informational queries (Broder 2002). More recently, Geng et al. (2008) proposed a version of *local learning* (Bottou and Vapnik 1992) for ranking: at test

time, a model is built using the only most similar queries to the test query. This is of course an expensive approach and the authors proposed offline approximations. Finally, Bian et al. (2010) proposed to learn specialized ranking functions for each query class and this learning is done jointly in the RankSVM framework. Let us give some more details of this approach, but for simplicity, let us consider the squared loss instead of a pairwise preference loss as in RankSVM. The objective function to be optimized in Bian et al. (2010) can then be written as:

$$\underset{F^1,\ldots,F^T}{\operatorname{argmin}} \sum_{q=1}^{Q} \sum_{i \in D_q} \left(\sum_{j=1}^{T} p_j^q F^j(x_i) - y_i \right)^2, \tag{18}$$

where q goes over the Q queries in the training set, D_q is the set of document indices associated with the q-th query, p_j^q is the probability that the query belongs to the j-th class ($\sum_j p_j^q = 1$) and finally, $F^1, \ldots, F^T$ are the specialized ranking functions corresponding to the T different query classes.

The potential drawback of the formulation (18) is that in the extreme case where each query belong to a only one category, that is when $p_j^q \in \{0, 1\}$, the estimation of the functions $F^1, \ldots, F^T$ becomes uncoupled. The problem of query dependent ranking is thus a case where multi-task learning can be helpful: a function F^0 captures the commonalities among the various query classes, while each of the F^j captures the specifics of a given query class. By introducing this global function F^0, we can rewrite (18) as

$$\underset{F^0,F^1,\ldots,F^T}{\operatorname{argmin}} \sum_{q=1}^{Q} \sum_{i \in D_q} \left(\sum_{j=1}^{T} p_j^q (F^0(x_i) + F^j(x_i)) - y_i \right)^2. \tag{19}$$

Instead of optimizing (19), we optimized a slightly different objective function, more precisely an upper bound on (19):

$$\underset{F^0,F^1,\ldots,F^T}{\operatorname{argmin}} \sum_{j=1}^{T} \sum_{q=1}^{Q} \sum_{i \in D_q} p_j^q (F^0(x_i) + F^j(x_i) - y_i)^2. \tag{20}$$

This is indeed an upper bound because:

$$\begin{aligned}
\left(\sum_{j=1}^{T} p_j^q (F^0(x_i) + F^j(x_i)) - y_i \right)^2 &= \left(\sum_{j=1}^{T} \sqrt{p_j^q} \sqrt{p_j^q} (F^0(x_i) + F^j(x_i) - y_i) \right)^2 \\
&\leq \sum_{j=1}^{T} p_j^q \sum_{j=1}^{T} p_j^q (F^0(x_i) + F^j(x_i) - y_i)^2 \\
&= \sum_{j=1}^{T} p_j^q (F^0(x_i) + F^j(x_i) - y_i)^2,
\end{aligned}$$

where the first and last equalities hold because $\sum_{j=1}^{T} p_j^q = 1$ and the inequality is the application of the Cauchy-Schwarz inequality.

The reason to consider (20) instead of (19) is two-fold:

1. The objective function (20) is more stringent because it aims at enforcing that the output of *every* function is near the target value while (19) only considers the weighted average output.
2. It is closer to our multi-boost framework. It can indeed be seen as an extension of multi-boost where all the training examples for a given task are identical, but where the weights for each task are different.

Multi-task boosting with objective function (20) is thus similar to algorithm 1, but with the following differences when building the regression trees:

– For task $j \geq 1$, the entire training set is used, the targets are as usual $F^0(x_i) + F^j(x_i) - y_i$, but each training sample is weighted by p_j^q.
– For task 0, the training samples are unweighted and the targets are $\sum_{j=1}^{T} p_j^q (F^0(x_i) + F^j(x_i) - y_i)$.

Evaluation To evaluate the effectiveness of multi-task learning for query dependent ranking, we considered queries and urls from our largest country (referred to M in Table 1). For training we had 45k queries and 887k urls, and for testing, 1132 queries and 39k urls. We then used an internal tool to compute the probability that a query belongs to one of these 5 predefined categories: *auto*, *local*, *product*, *travel*, and a *general* catch-all category. Since a query can belong to multiple categories, these probabilities do not necessarily sum to 1: we then normalized the probabilities over these 5 categories to ensure that they sum to 1.

Figure 6 plots the DCG-5 on the test set as a function of the total number of trees. The baseline ignores the different categories and trains a global model. For this comparison, all the hyperparameters for both methods have been set to the same value. We stopped the boosting when the number of trees for all classes are at least 1500, that is when $\min_{j \geq 1} \#\text{trees}(F^0) + \#\text{trees}(F^j) = 1500$. But the x-axis is the total number of trees: $\sum_{j=0}^{5} \#\text{trees}(F^j)$. This is the reason why the blue curve extends a bit beyond 1500 in Fig. 6.

The performance until 1000 trees is similar for both methods, which is understandable because in earlier iterations, the multi-task method takes mostly global steps and the commonalities between the tasks is first modeled. But after 1000 trees, the baseline saturates,

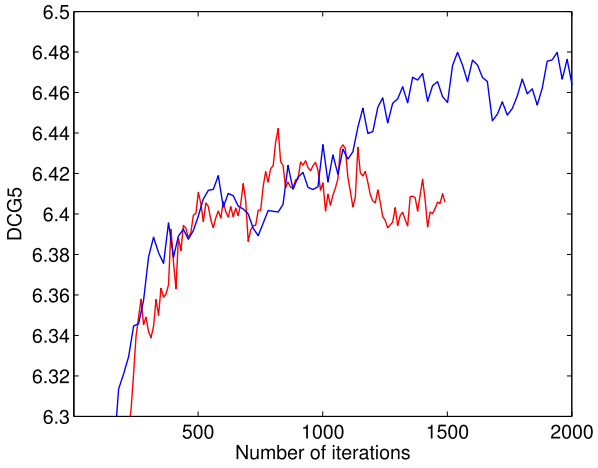

Fig. 6 (Color online) DCG-5 as a function of the number trees for query dependent ranking. The *blue curve* corresponds to the multi-task model (20) trained over the $T = 5$ categories, while the *red curve* is the equivalent of a *pooling* baseline where the categories are ignored

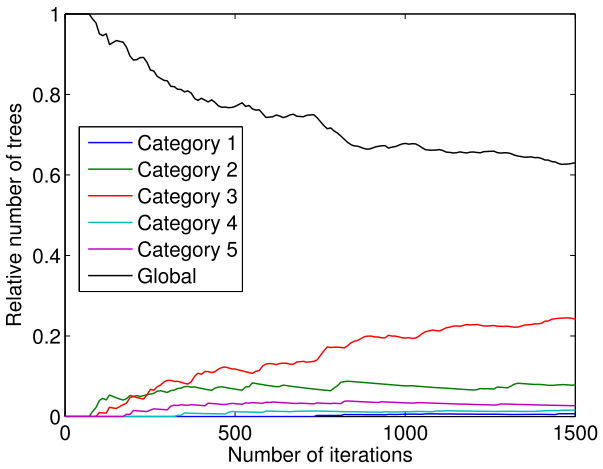

Fig. 7 (Color online) Percentage of time the global function F^0 and the local functions F^j, $j \geq 1$ have been updated as a function of the number of iterations

while the multi-task keeps improving. It is indeed able to model the specifics of each query class better. This results in a DCG improvement of 0.74% when the number of trees is selected optimally for both methods. Finally, Fig. 7 shows the fraction of times each F^j has been updated.

9 Multi-task regression on a public dataset

To compare our multi-boost with many other existing approaches to multi-task learning, we applied our algorithm on the *school* dataset. This dataset consists of 15362 examination records of students from 139 different schools. Each school serves as a task and the aim is to predict the grades of students based on socio-economic, demographic and other features. In fact, this dataset has been used extensively in the multi-task learning literature. Further, ten train-test splits of the data that were previously used by Bakker and Heskes (2003), Evgeniou and Pontil (2004) are also publicly available. Since our results are also on these ten splits, a direct comparison with the previous literature is thus possible.

The metric used in evaluating the multi-task approaches in Bakker and Heskes (2003), Evgeniou and Pontil (2004), Argyriou et al. (2008) is the so-called "explained-variance" which is a percentage version of the well known measure R^2 in statistics; we evaluated our approach with the same metric.

For each data fold, we first divided the training data into two splits. A two fold internal validation (among these two splits) was done to pick the number of iterations of boosting and ϵ. In this case, we merely used a one node regression tree as a weak learner. With the selected values of ϵ and the number of boosting iterations, we learned a regression function using the entire training data in each fold. We then obtained the explained-variance on the test data (for all the tasks put together). To get a better perspective, we also obtained results for pooling (same as our boosting but only global steps allowed) and independent tasks. In the case of independent tasks, the values of the parameters were tuned for each task independently, yet the final explained-variance was by putting together the predictions for all the tasks.

The results are summarized in Table 8. It can be observed that pooling improves over independent tasks and multi-boost improves further over pooling. A direct comparison with results in the previous literature (Bakker and Heskes 2003; Evgeniou and Pontil 2004; Argyriou et al. 2008) shows that our results are comparable to the state-of-the-art.

Table 8 Mean and standard deviation of explained-variance on school dataset obtained by various boosting algorithms and previous approaches (appear with citations). Results obtained by multi-boost algorithm is comparable to the best result in the literature

Method	Explained-variance
Independent	34.1 ± 1.2
Pooling—weighted	35.8 ± 1.2
Pooling—unweighted	36.1 ± 1.1
Bayesian MTL (Bakker and Heskes 2003)	29.5 ± 0.4
Regularized MTL (Evgeniou and Pontil 2004)	34.8 ± 0.5
MTL-FEAT (linear kernel) (Argyriou et al. 2008)	37.1 ± 1.5
MTL-FEAT (Gaussian kernel) (Argyriou et al. 2008)	37.6 ± 1.0
Multi-boost—weighted	37.3 ± 1.2
Multi-boost—unweighted	37.7 ± 1.2

10 Conclusions

In this paper we introduced a novel multi-task learning algorithm for gradient boosted decision trees and applied it to web search ranking. We mainly focused on the problem of learning the ranking functions for various countries in a multi-task learning framework. The customization of the ranking models to each country happens naturally by modeling both the characteristics of the local countries and the commonalities separately. We provided a thorough evaluation of multi-task web search ranking on large scale real world data. Our multi-task learning method lead to reliable improvements in DCG, especially after specifically selecting sub-sets that are learned jointly.

As future work, we could look into modeling more diverse interactions between different countries. In particular, our approach is by no means restricted to a single global model. In fact, given domain-knowledge, it could make sense to add additional functions that capture the interactions between a sub-set of the countries. For example, one could imagine functions $F^{T+1}, \ldots, F^{T+c}$ for various continents, such that the relevance of a document of task t, from continent c, is predicted with $F^0(x) + F^t(x) + F^{T+c}(x)$. Even finer interactions could be modeled through hierarchies of additional functions.

Even though our primary interest was to learn ranking functions across countries, our framework is very general and can be applied to other machine learning tasks beyond ranking for which boosted decision trees are well suited. We illustrated this point by applying our algorithm to a standard multi-task benchmark dataset.

Appendix

We derive here the optimal solution of optimization problem (13):

$$\min_{\Delta\beta^t} \sum_{t=0}^{T} \left\langle \Delta\beta^t, g^t \right\rangle \quad \text{s.t.} \quad \sum_{t=0}^{T} \lambda_t \|\Delta\beta^t\|_1 \leq \epsilon.$$

With a change of variables $\Delta\tilde{\beta}_j^t \leftarrow \lambda_t \Delta\beta_j^t$, the problem becomes one of minimizing an inner product under ℓ_1 constraint:

$$\min_{\Delta\tilde{\beta}} \left\langle \Delta\tilde{\beta}, \tilde{g} \right\rangle \quad \text{s.t.} \quad \|\Delta\tilde{\beta}\|_1 \leq \epsilon,$$

where we came back to the stacked representation (7) and defined $\tilde{g}_j^t = g_j^t/\lambda_t$.

Let $\bar{t}$ and $\bar{j}$ be the indices such that

$$\bar{t}, \bar{j} = \operatorname*{argmin}_{t,j} \tilde{g}_j^t. \tag{21}$$

Note that

$$\min_{t,j} \tilde{g}_j^t = -\max_{t,j} \tilde{g}_j^t = -\max_{t,j} |\tilde{g}_j^t|. \tag{22}$$

This is because of the assumption that $\mathcal{H}$ is closed under negation and thus,

$$\forall t, \forall j, \exists k, \quad g_j^t = -g_k^t.$$

A lower bound on the dot product can be found using Hölder's inequality:

$$\begin{aligned} \langle \Delta\tilde{\beta}, \tilde{g} \rangle &\geq -|\langle \Delta\tilde{\beta}, \tilde{g} \rangle| \\ &\geq -\|\Delta\tilde{\beta}\|_1 \, \|\tilde{g}\|_\infty \\ &\geq \epsilon \, \tilde{g}_{\bar{j}}^{\bar{t}}, \end{aligned}$$

where we made use of (21) and (22) for the last inequality.

Let $\Delta\tilde{\beta}_j^t = \epsilon$ if $j = \bar{j}$ and $t = \bar{t}$, 0 otherwise. This $\Delta\tilde{\beta}$ satisfies the ℓ_1 constraint and is also optimal because it satisfies the above lower bound with equality.

Coming back to the original variables, the optimal $\Delta\beta$ is thus given by (14).

References

Argyriou, A., Evgeniou, T., & Pontil, M. (2007). Multi-task feature learning. In *Advances in neural information processing systems* (Vol. 19). Cambridge: MIT Press.

Argyriou, A., Evgeniou, T., & Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, *73*(3), 243–272.

Bakker, B., & Heskes, T. (2003). Task clustering and gating for Bayesian multitask learning. *Journal of Machine Learning Research*, *4*, 83–99. doi:10.1162/153244304322765658.

Ben-David, S., & Schuller, R. (2003). Exploiting task relatedness for multiple task learning. In *16th annual conference on learning theory* (pp. 567–580).

Bian, J., Li, X., Li, F., Zheng, Z., & Zha, H. (2010). Ranking specialization for web search: a divide-and-conquer approach by using topical ranksvm. In *WWW'10: proceedings of the 19th international World Wide Web conference*.

Bottou, L., & Vapnik, V. (1992). Local learning algorithms. *Neural Computation*, *4*(6), 888–900.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. London/Boca Raton: Chapman & Hall/CRC.

Broder, A. (2002). A taxonomy of web search. *SIGIR Forum*, *36*(2), 3–10.

Burges, C., Shaked, T., Renshaw, E., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. In *Internation conference on machine learning* (pp. 89–96).

Cao, Y., Xu, J., Liu, T., Li, H., Huang, Y., & Hon, H. (2006). Adapting ranking SVM to document retrieval. In *Proceedings of the 29th international ACM SIGIR conference on research and development in information retrieval* (pp. 186–193).

Caponnetto, A., Micchelli, C., Pontil, M., & Ying, Y. (2008). Universal multi-task kernels. *The Journal of Machine Learning Research*, *9*, 1615–1646.
Caruana, R. (1997). Multitask learning. In *Machine learning* (pp. 41–75).
Chapelle, O., & Wu, M. (2010). Gradient descent optimization of smoothed information retrieval metrics. *Information Retrieval Journal*, *13*(3), 216–235.
Chen, D., Xiong, Y., Yan, J., Xue, G. R., Wang, G., & Chen, Z. (2009). Knowledge transfer for cross domain learning to rank. Information Retrieval.
Collobert, R., & Weston, J. (2008). A unified architecture for NLP: deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167). New York: ACM.
Cossock, D., & Zhang, T. (2006). Subset ranking using regression. In *Proceedings of the conference on learning theory*.
Evgeniou, T., & Pontil, M. (2004). Regularized multi-task learning. In *KDD* (pp. 109–117).
Evgeniou, T., Micchelli, C., & Pontil, M. (2006). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, *6*(1), 615.
Friedman, J. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, *29*, 1189–1232.
Gao, J., Wu, Q., Burges, C., Svore, K., Su, Y., Khan, N., Shah, S., & Zhou, H. (2009). Model adaptation via model interpolation and boosting for web search ranking. In *EMNLP* (pp. 505–513).
Geng, X., Liu, T. Y., Qin, T., Arnold, A., Li, H., & Shum, H. Y. (2008). Query dependent ranking using k-nearest neighbor. In *SIGIR'08: proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 115–122). New York: ACM.
Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. In A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (Eds.), *Advances in large margin classifiers* (pp. 115–132). Cambridge: MIT Press.
Jarvelin, K., & Kekalainen, J. (2002). IR evaluation methods for retrieving highly relevant documents. In *ACM special interest group in information retrieval (SIGIR)* (pp. 41–48). New York: ACM.
Jebara, T. (2004). Multi-task feature and kernel selection for svms. In *Proceedings of the 21st international conference on machine learning*.
Kang, I. H., & Kim, G. (2003). Query type classification for web document retrieval. In *SIGIR'03: Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 64–71). New York: ACM.
Li, P., Burges, C., & Wu, Q. (2008). Mcrank: learning to rank using multiple classification and gradient boosting. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in neural information processing systems* (Vol. 21, pp. 897–904). Cambridge: MIT Press.
Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Boosting algorithms as gradient descent in function space. In *Neural information processing systems* (Vol. 12, pp. 512–518).
Maurer, A. (2006). Bounds for linear multi-task learning. *Journal of Machine Learning Research*, *7*, 117–139.
Rosset, S., Zhu, J., Hastie, T., & Schapire, R. (2004). Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, *5*, 941–973.
Schölkopf, B., & Smola, A. (2002). *Learning with kernels*. Cambridge: MIT Press.
Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008). SoftRank: optimizing non-smooth rank metrics. In *Proceedings of the 1st ACM international conference on web search and data mining* (pp. 77–86).
Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In D. Touretzky & M. Mozer (Eds.), *Advances in neural information processing systems (NIPS)* (Vol. 8, pp. 640–646). Cambridge: MIT Press.
Wang, X., Zhang, C., & Zhang, Z. (2009). Boosted multi-task learning for face verification with applications to web image and video search. In *Proceedings of IEEE computer society conference on computer vision and patter recognition*.
Weinberger, K., Dasgupta, A., Attenberg, J., Langford, J., & Smola, A. (2009). Feature hashing for large scale multitask learning. In *ICML*.
Xue, Y., Liao, X., Carin, L., & Krishnapuram, B. (2007). Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, *8*, 2007.
Yu, K., Tresp, V., & Schwaighofer, A. (2005). Learning Gaussian processes from multiple tasks. In *Proceedings of the 22nd international conference on machine learning*.
Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A support vector method for optimizing average precision. In *Proceedings of the 30th international ACM SIGIR conference on research and development in information retrieval* (pp. 271–278).
Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Chen, K., & Sun, G. (2008). A general boosting method and its application to learning ranking functions for web search. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in neural information processing systems* (Vol. 20, pp. 1697–1704).