Multi-Task Learning for Boosting with Application to Web Search Ranking

Olivier Chapelle
Yahoo! Labs
Sunnyvale, CA
chap@yahoo-inc.com

Pannagadatta Shivaswamy
Dept of Computer Science
Columbia University, NY
pks2103@cs.columbia.edu

Srinivas Vadrevu
Yahoo! Labs
Sunnyvale, CA
svadrevu@yahoo-inc.com

Kilian Weinberger
Washington University
Saint Louis, MO
kilian@wustl.edu

Ya Zhang
Shanghai Jiao Tong University
Shanghai, China
yazhang@sjtu.edu.cn

Belle Tseng
Yahoo! Labs
Sunnyvale, CA
belle@yahoo-inc.com

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 the specifics of each learning task with task-specific parameters and the commonalities between them through shared parameters. This enables implicit data sharing and regularization. We evaluate our learning method on web-search ranking data sets from several countries. Here, multitask learning is particularly helpful as data sets from different countries vary largely in size because of the cost of editorial judgments. Our experiments validate that learning various tasks jointly can lead to significant improvements in performance with surprising reliability.

Categories and Subject Descriptors

I.2.6 [**Artificial intelligence**]: Learning; H.3.3 [**Information storage and retrieval**]: Information search and retrieval

General Terms

Algorithms

1. INTRODUCTION

Multi-task learning algorithms [2] aim to improve the performance of several learning tasks through shared models. Previous work focussed primarily on neural networks, k-nearest neighbors [2] and support vector machines [6]. In this paper, we introduce a novel multi-task learning algorithm for gradient boosting. This is motivated by our interest in web search ranking: gradient boosted decision trees are indeed among the state-of-the-art algorithms for large-scale web-search ranking [11, 19].

Web search ranking is often treated as a supervised machine learning problem [12]: 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 indeed specialized rankings for different countries or regions. The problem of high editorial cost is 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 countries 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.

A common related line of research 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.* [8] address this particular case with boosted decision trees through model interpolation and refinement.

Related to our work, but fundamentally different, is multi-task boosting for face verification [17]. It learns a set of boosted classifiers and is based on a probabilistic model where a multinomial variable indicates how much each boosted classifier contributes to each task. The learning algorithm involves Expectation-Maximization (EM) to learn both the multinomial random variables as well as the classifiers. Dai *et al.* [5] developed a variation of AdaBoost [15] that can incorporate training data from a different distribution than the test set. Instead of learning multiple models, their approach down-weights data points that are not representable for the test set. A different method of cross domain learning is discussed in [3], which uses both feature level and instance level knowledge transfer across domains. The feature level knowledge transfer is modeled as an optimization function in RankingSVM [9] by learning the underlying common feature representation across domains. The instance level knowledge transfer is achieved by sample weighting.

Compared to all these approaches, 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 [14]. To the best of our knowledge we are not aware of any work that jointly models several tasks by explicitly learning both the commonalities and idiosyncrasies through gradient boosted regression.

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.

Although we mainly focus on multi-task learning for ranking across different countries, it is important to point out that our algorithm can readily be used in other scenarios. In the context of web-search one might encounter different multi-task problems, such as customizing ranking functions for different query types by modeling both the commonalities across the query types and the idiosyncrasies of each query set. Further, our algorithm is not specific to web search ranking and is equally applicable to any standard machine learning task such as classification or regression.

The rest of the paper is organized as follows. We formally introduce the multi-task learning problem in section 2 and propose our approach in section 3. The connection between boosting and ℓ_1 regularization serves as motivation for our derivations. In sections 4, 5 and 6, we evaluate our method on several real-world large scale web-search ranking data sets. Section 7 presents the conclusions and future directions for our work.

2. BACKGROUND

Notation and setup.

Assume that we are given learning tasks $t \in \{1, 2, \dots, T\}$. Further, the data for these tasks, $\{(x_1, y_1), \dots, (x_n, y_n)\}$, 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 $\{1, \dots, n\}$ (*i.e.*, $I^t \cap I^s = \emptyset$ when $t \neq s$ and $\cup_{t=1}^T I^t = \{1, \dots, n\}$). We also define $I^0 = \{1, \dots, n\}$. 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 regression setting, we might consider squared loss:

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

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,

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

Previous work.

Previous work has mainly focused on Neural Networks [2, 4] or Support Vector Machines [6]. This latter work is of particular interest because it inspired the algorithm presented in this paper. In this SVM based multi-task learning, a classifier w_t is specifically dedicated for the 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:

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

In [6], 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

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

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 λ_0 is small and $\lambda_i \to +\infty$, $\forall i \geq 1$, 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* [16] 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 explicitly 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 [6]. 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, the authors of [18] 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.

3. 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.

3.1 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 $\phi : \mathcal{X} \rightarrow \mathbb{R}^J$ 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 [13]. 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 $\mathbb{R}^J$ as β^t. As J can be very large, we can only store vectors $\beta \in \mathbb{R}^J$ 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

$$\min_{\beta^0,\beta^1,\ldots,\beta^T} \sum_{t=1}^T C^t(\beta^0 + \beta^t) \quad \text{s.t.} \quad \sum_{t=0}^T \lambda_t \|\beta^t\|_1 \leq \mu, \tag{6}$$

where, as in (2), $C^t(\beta)$ is defined as $C^t(\ldots, \langle \beta, \phi(x_i)\rangle, \cdots)_{i\in I_t}$. 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 $\mathbb{R}^J$ 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 [1] – 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).

3.2 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 [14] and ensure that our solver is in fact an instance of gradient boosting [13]. 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 $\beta \in \mathbb{R}^{J(T+1)}$, defined as

$$\beta = [\beta^{0\top}, \ldots, \beta^{T\top}]^\top,\ C(\beta) := \sum_{t=1}^T C^t(\beta^0 + \beta^t). \tag{7}$$

This reduces (6) to the much simpler 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 [14] refer to as $\epsilon - boosting$. 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. 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 [14], 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 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 this ϵ update is performed.

Multitask 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

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

with $g^t_j := \dfrac{\partial C}{\partial \beta^t_j}$.

Let us define the outputs at the training points as

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

Using the chain-rule,

$$g^t_j = \sum_{i=1}^n \frac{\partial C(u)}{\partial u_i} \frac{\partial u_i}{\partial \beta^t_j}.$$

On the other hand,

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

Combining the above equations, we finally obtain:

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

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

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

With a change of variables $\tilde{\beta}_j^t \leftarrow \lambda_t \beta_j^t$, the problem becomes one of minimizing an inner product under ℓ_1 constraint. 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 easy to show that the solution of (13) is given by:

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

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

It remains to show that we can compute the single nonzero 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, i.e. $\sum_{i \in I^t} h(x_i)^2 = 1$[1], we can express (5) with $z_i = -\frac{\partial C(u)}{\partial u_i}$ as,

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

The optimal couple (t,j) from (14) is thus $(\hat{t}, \hat{\jmath}(\hat{t}))$ with

$$\hat{t} = \underset{t}{\text{argmax}} \frac{1}{\lambda_t} \sum_{i \in T^t} h_{\hat{\jmath}(t)}(x_i) z_i. \tag{15}$$

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 algorithm *multi-boost* with the update-rule (16) is summarized in Algorithm 1 and is illustrated in Figure 1. The computational complexity of this algorithm is linear in the number of training samples as well as in the number of tasks.

3.3 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 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 equation (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.

[1]There is a flaw in our reasoning since in general this equation cannot hold for all t simultaneously. However, we will later present an extension where the functions h depend on t. That will solve this problem.

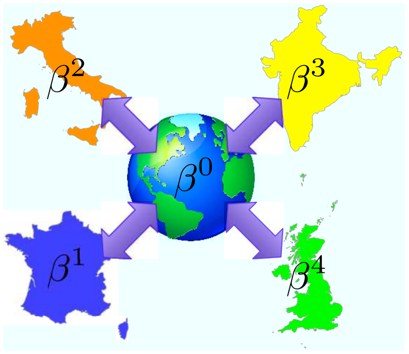

Figure 1: A layout of four ranking tasks that are learned jointly. The four countries symbolize the different ranking functions that need to be learned, where $\beta^1, \ldots, \beta^4$ are the parameter vectors that store the specifics of each individual task. The various tasks interact through the joint model, symbolized as a globe with parameter vector β^0.

Second order information.

Instead of performing an ϵ-gradient step, we follow the framework of [19] 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.

Weights.

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

4. EXPERIMENTAL SETUP

In this section we present the experimental setup for our experiments. The data sets include large-scale web ranking training sets for various countries. All the data sets contain randomly sampled queries from the search engine query logs.

	Examples			Queries		
Country	Train	Valid	Test	Train	Valid	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	–	–

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.

For each query, a set of URLs is sampled from the results retrieved by several search engines. For each query-url pair, an editorial grade containing $0-4$ is obtained that describes the relevance of the url to the query. Each query-url pair is represented with several hundred features.

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.

The performance of various ranking models is measured on the test set using Discounted Cumulative Gain [10] at fifth position, which we refer to as DCG-5. We experimented with two different types of loss functions: the squared loss as in GBDT [7] and a pairwise preference loss as in RankSVM [9] and GBRank [19]: if x_i is to be preferred to x_j, the corresponding loss for that pair is $\max(0, 1-(f(x_i)-f(x_j)))^2$. We thus refer to GBDT and GBRank as our learning methods for ranking both with and without multi-task learning. More background information on learning to rank can be found in a recent survey paper [12].

The experimental results have been divided into two sections. In the next section we present preliminary results on a small subset of the complete feature set. For the large-scale experiments in Section 6, we present the results with the complete feature set containing more than 500 features such as text match scores, spam scores, link based features, click features and page classifier outputs. 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 [7].

5. PRELIMINARY EXPERIMENTS

In this section we present preliminary experimental results on a subset of data sets with a smaller 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.

Also 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 – were 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 Figure 2. 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 slighlty 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.

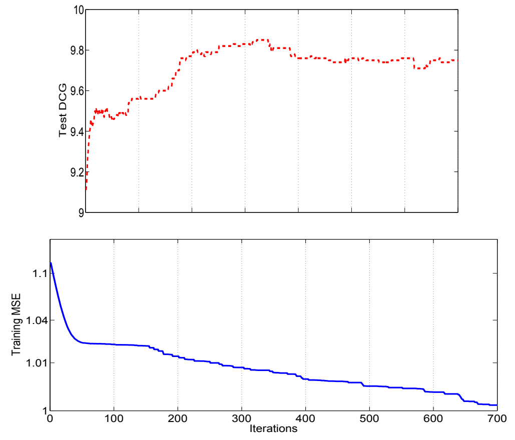

Figure 2: Train MSE and test DCG as a function of the number of iterations.

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

- **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;

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

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

- **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 Section 3.3, 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 inverse 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.[2]

Steps taken by the two weighting schemes: Typical behavior of the steps taken with the two weighting schemes are shown in Figure 3. In both 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 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.

[2]What we mean here is that improvements reported in web search ranking papers are typically of the order of 1%; see for instance Figure 1 of [19].

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

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

6. 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 Sections 6.1, 6.2 and 6.3, 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 Section 6.4 and 6.5, we did complete model selection on the validation set and report the results on the test set.

Note that we have two different experimental setting 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 Sections 6.4 and 6.5, we present results based on *web-scale* experimental setting, where all the documents[3] 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.

[3]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.

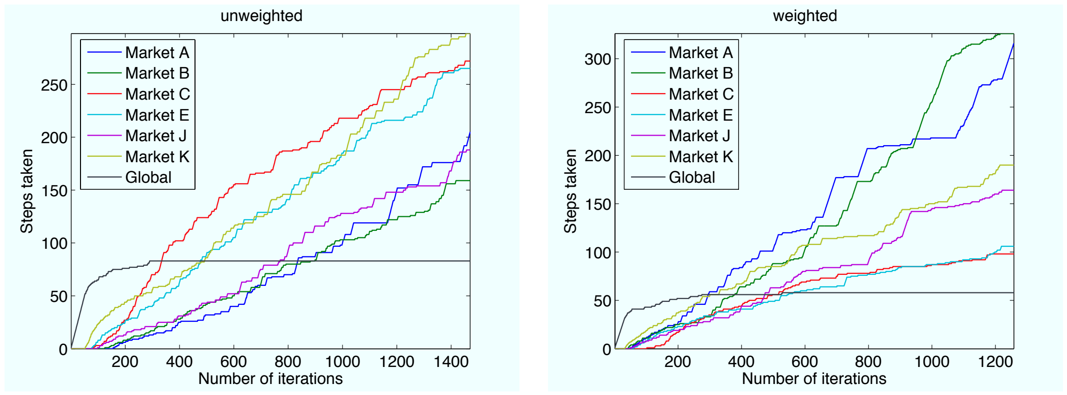

Figure 3: Steps taken by the multi-task algorithm with the two weighting schemes. The country labels A-K are sorted by increasing data set sizes. The *unweighted* (left) version takes more steps for the larger data sets, whereas the *weighted* (right) variation focusses more on countries with less data.

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

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.

6.1 Effect of the loss function

As indicated earlier, our method can be applied to a any loss function and in particular it can be combined with a pairwise ranking model (see Section 4). 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 of both learning algorithms with the two weighting schemes discussed in Section 5. It can be seen that in both pointwise and pairwise ranking schemes, *multitask* ranking models have better average performance over the *independent* models.

6.2 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 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 for each country 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.

6.3 Comparison over adaption models

In this section we present comparison results of our *multi-boost* method with the domain adaptation method described in [8]. 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 4 shows the DCG-5 gains of our method over the adaptation method in [8]. 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 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.

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		†						†

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.

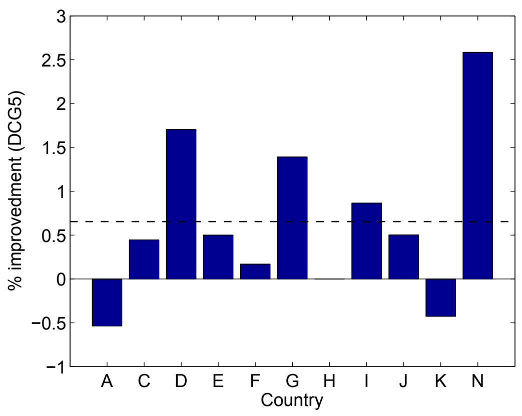

Figure 4: DCG-5 gains of multi-task models over the adaption models. The dashed line represents the mean gain, 0.65%.

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 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.

6.4 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 [7]. 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.

6.5 Experiments with Global Models

As discussed in Section 3, 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

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

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

of the multi-task global models over that baseline ranking model F^0. 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.

7. CONCLUSIONS

In this paper we introduced a novel multi-task learning algorithm based on gradient boosted decision trees that is specifically designed with web search ranking in mind. We model the problem of learning the ranking model 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.

The results in this paper validate that multi-task learning has a natural application in web-search ranking. As future work, we want to apply this multi-task approach to other ranking adaptation problems where the tasks could involve various query types such as navigational and informational queries. Our proposed multi-task learning framework could learn a ranking model for all of these query types jointly while still learning a generic global learning model that can work across all types of queries.

Furthermore, we could apply our algorithm to other machine learning tasks beyond ranking for which boosted decision trees are well suited.

8. REFERENCES

[1] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees.* Chapman & Hall/CRC, 1984.

[2] R. Caruana. Multitask learning. In *Machine Learning*, pages 41–75, 1997.

[3] D. Chen, Y. Xiong, J. Yan, G.-R. Xue, G. Wang, and Z. Chen. Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 2009.

[4] R. Collobert and J. Weston. A unified architecture for NLP: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM New York, NY, USA, 2008.

[5] W. Dai, Q. Yang, G. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.

[6] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *KDD*, pages 109–117, 2004.

[7] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[8] J. Gao, Q. Wu, C. Burges, K. Svore, Y. Su, N. Khan, S. Shah, and H. Zhou. Model adaptation via model interpolation and boosting for web search ranking. In *EMNLP*, pages 505–513, 2009.

[9] R. Herbrich, T. Graepel, and K. Obermayer. *Large margin rank boundaries for ordinal regression*, pages 115–132. MIT Press, Cambridge, MA, 2000.

[10] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR*, pages 41–48. New York: ACM, 2002.

[11] P. Li, C. J. C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *NIPS*, 2007.

[12] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–231, 2009.

[13] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent in function space. In *Neural information processing systems*, volume 12, pages 512–518, 2000.

[14] S. Rosset, J. Zhu, T. Hastie, and R. Schapire. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

[15] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.

[16] B. Schölkopf and A. Smola. *Learning with kernels.* MIT press Cambridge, MA, 2002.

[17] X. Wang, C. Zhang, and Z. Zhang. 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*, 2009.

[18] K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford, and A. Smola. Feature hashing for large scale multitask learning. In *ICML*, 2009.

[19] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *NIPS*, 2007.