4-2011

Weight-based boosting model for cross-domain relevance ranking adaptation

Peng CAI

Wei GAO
Singapore Management University, weigao@smu.edu.sg

Kam-Fai WONG

Aoying ZHOU

Weight-Based Boosting Model for Cross-Domain Relevance Ranking Adaptation*

Peng Cai[1], Wei Gao[2], Kam-Fai Wong[2,3], and Aoying Zhou[1]

[1] East China Normal University, Shanghai, China
pengcai2010@gmail.com, ayzhou@sei.ecnu.edu.cn
[2] The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{wgao,kfwong}@se.cuhk.edu.hk
[3] Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing, China

Abstract. Adaptation techniques based on importance weighting were shown effective for RankSVM and RankNet, viz., each training instance is assigned a *target weight* denoting its importance to the target domain and incorporated into loss functions. In this work, we extend RankBoost using importance weighting framework for ranking adaptation. We find it non-trivial to incorporate the target weight into the boosting-based ranking algorithms because it plays a contradictory role against the innate weight of boosting, namely *source weight* that focuses on adjusting source-domain ranking accuracy. Our experiments show that among three variants, the *additive* weight-based RankBoost, which dynamically balances the two types of weights, significantly and consistently outperforms the baseline trained directly on the source domain.

1 Introduction

Learning to rank [4] is to derive effective relevance ranking functions based on a large set of human-labeled data. Boosting has been extensively studied for learning to rank [1,2,8,9]. However, existing ranking algorithms, including the boosting-based ones, are only proven effective for data from the same domain. In real applications, it is prohibitive to annotate training data for every search domain. Ranking performance may suffer when the training and test have to take place on different domains.

A promising direction is to learn a cross-domain adaptation model for ranking. Two key problems should be resolved: (1) how to measure the relatedness of two domains appropriately; (2) how to utilize this information in ranking algorithms for adaption. [3] adopted a classification hyperplane to derive the importance weight of documents in the source domain that reflects their similarity to the target domain and is then incorporated into the rank loss function.

When applying this method, we find it non-trivial to integrate importance weight into boosting-based algorithms such as RankBoost [2]. The reason is

* This work is partially supported by NSFC grant (No. 60925008), 973 program (No. 2010CB731402) and 863 program (No. 2009AA01Z150) of China.

P. Clough et al. (Eds.): ECIR 2011, LNCS 6611, pp. 562–567, 2011.

that these algorithms bear an inherently weight-based exponential loss and this innate weight in the loss function (*source weight*) plays a contradictory role with the importance weight introduced for adaptation purpose (*target weight*). An appropriate balance must be made between these two types of weight; otherwise adaptation may fail since the model can easily overfit source data due to the great impact of source weight on weak rankers selection.

In this work, we develop three Weight-based RankBoost (WRB) algorithms to balance the source and target weights, namely expWRB, linWRB and additive WRB (addWRB). The first two methods incorporate the target weight in straightforward and static ways, and the third combines the weights from a *global* perspective based on a forward stage-wise additive approach [6] to achieve a dynamic tradeoff. Our results demonstrate that addWRB consistently and significantly outperforms the baseline trained directly on the source domain.

2 Target Weight and Source Weight

2.1 Source Weight

RankBoost [2] aims to find a ranking function F to minimize the number of misordered document pairs. Given document pairs $\{\langle x_i, x_j\rangle\}$, the ranking loss is defined as $rLoss(F) = \sum_{i,j} W(x_i, x_j) I(F(x_i) \geq F(x_j))$, where $W(x_i, x_j)$ is the source weight distribution, $I(.)$ is an binary indicator function, and $F(x_i) \geq F(x_j)$ suggests that the ranking function assigns a higher score to x_i than to x_j while the ground truth is x_i has a lower rating than x_j. At each round of training, $W(.)$ is updated for the next round to focus on those misordered pairs. The update formula in round t is given as follows:

$$W_{t+1} = \frac{1}{Z_t} W_t(x_i, x_j) \exp(\alpha_t (f_t(x_i) - f_t(x_j))) \tag{1}$$

where $f_t(x)$ is the 0-1 valued weak ranker derived from a ranking feature x, α_t is the coefficient of f_t so that $F = \sum_t \alpha_t f_t(x)$, and Z_t is normalization factor. Inherently, source weight is designed to control the selection of weak rankers to minimize ranking errors in source domain.

2.2 Target Weight

In ranking adaptation, the knowledge of relevance judgement should be strengthened on those documents that are similar to the target domain so that the learning can be focused on correctly ranking these important documents. [3] used the cross-domain similarity to transfer ranking knowledge. The distance of a source-domain document to the classification hyperplane was calculated as target weight to measure the importance of the document. Then the pointwise weight was converted to pairwise for compatible with the popular pairwise approach (see [3] for details). The general loss term was extended as follows:

$$\sum_{i,j} w(x_i, x_j) * rLoss_{ij}(.) \tag{2}$$

where $rLoss_{ij}(.)$ is the pairwise loss and $w(x_i, x_j)$ is the target weight.

Algorithm 1. expWRB–Weighted RankBoost with target weight inside the exponent

Input: Query-document set of source domain; Target weights of M document pairs $\{w(x_i, x_j)\}_1^M$ based on the ground truth.
Output: Ranking function $F(x)$.
1. Initialize $W_1(x_i, x_j) = \frac{1}{M}$ for all i, j;
2. **for** $t = 1$; $t \leq T$; $t++$ **do**
3. Select weak ranker $f_t(x)$ using W_t and w;
4. Set coefficient α_t for $f_t(x)$;
5. For each (x_i, x_j), update source weight using $W_{t+1} = \frac{1}{Z_t} W_t(x_i, x_j) \exp(\alpha_t w(x_i, x_j)(f_t(x_i) - f_t(x_j)))$;
6. **end for**
7. **return** $F(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$

3 Weight-Based Boosting Models for Ranking Adaptation

In standard RankBoost, the source weight is updated iteratively so that the weak rankers can focus on those misordered pairs with large source weight that commonly reside near the decision boundary deemed more difficult to order. However, these pairs may be unnecessarily important to the target domain. Meanwhile, for those misordered pairs with low source weight, even though they contain some important cross-domain ranking knowledge (i.e., having high target weight), the algorithm does not prioritize to correct their ranking. The two types of weight play contradictory roles and must be appropriately balanced.

The objective of our adaptive RankBoost is to minimize the weighted ranking loss $wLoss(F) = \sum_{i,j} W(x_i, x_j) I(F(x_i) \geq F(x_j)) w(x_i, x_j)$ following Eq. 2. There are two straightforward ways to incorporate the target weight into the source weight's update formula (Eq. 1):

$$W_{t+1} = \frac{1}{Z_t} W_t(x_i, x_j) \exp(\alpha_t \boxed{w(x_i, x_j)} (f_t(x_i) - f_t(x_j))), \tag{3}$$

$$W_{t+1} = \frac{1}{Z_t} W_t(x_i, x_j) \boxed{w(x_i, x_j)} \exp(\alpha_t (f_t(x_i) - f_t(x_j))). \tag{4}$$

Thus, we obtain two versions of Weight-based RankBoost (WRB), namely expWRB corresponding to the target weight inside the exponent (Eq. 3) and linWRB corresponding to the linearly combined target weight (Eq. 4).

3.1 expWRB

The procedure of expWRB is shown as Algorithm 1. Other than the updating of source weight in step 5, expWRB also differs from standard RankBoost in step 3 where both source and target weights are used to search for the objective function F to minimize the weighted rank loss.

In each round t, we can choose an appropriate α_t and $f_t(x)$ to minimize Z_t in step 3. Z_t is minimized by maximizing $r_t = \sum_{i,j} W_t(x_i, x_j) w(x_i, x_j)(f_t(x_j) - f_t(x_i))$ and set $\alpha_t = \frac{1}{2} \ln \frac{1+r_t}{1-r_t}$ in step 4 [2].

3.2 linWRB

Replacing the updating rule in step 5 in Algorithm 1 with Eq. 4, we can obtain linWRB with linearly combined target weight. Similarly, we minimize Z_t for minimizing the weighted loss in each round following [2]. Given a binary weak ranker $f_t(x) \in \{0, 1\}$ and $a \in \{-1, 0, +1\}$, we set $R_a = \sum_{i,j} W(x_i, x_j) w(x_i, x_j) I(f_t(x_i) - f_t(x_j) = a)$. Then $Z_t = R_{+1} \exp(\alpha_t) + R_0 + R_{-1} \exp(-\alpha_t)$. Z_t is minimized by setting $\alpha_t = \frac{1}{2} \ln \frac{R_{-1}}{R_{+1}}$. The weak ranker f_t is selected when Z_t is minimal.

3.3 Additive Weight-Based RankBoost

Standard RankBoost updates source weight in the round $t+1$ based on the current weak ranker f_t *locally* (see Eq. 1). A better way is to calculate W_{t+1} *globally* using the ensemble function F_t that combines all the weak rankers learned up to the current round like an additive approach [6]. The update rule is given by $W_{t+1} = \frac{1}{Z_t} \exp(F_t(x_i) - F_t(x_j))$ where $F_t(x) = F_{t-1}(x) + \alpha_t f_t(x)$, which eliminates the previous round source weight. Then the target weight can be incorporated straightforwardly as $W_{t+1} = \frac{1}{Z_t} w(x_i, x_j) \exp(F_t(x_i) - F_t(x_j))$.

However, the model easily overfits the source domain due to the great impact on the updating of source weight from the exponential term. We introduce a scaling factor λ to adjust the source weight dynamically. The idea is that we update λ considering ranking difficulty measured by the proportion of correctly ordered pairs in each round:

$$\lambda_t = \lambda_{t-1} * \frac{\#\text{ of correctly ordered pairs by } F_t}{\text{Total } \#\text{ of pairs to rank}} \tag{5}$$

In a difficult task where wrong pairs dominate, λ decreases quickly and cancel out the exponential growth of source weight so that target weight can affect weak ranker selection properly.

Based on this intuition, we propose the Additive Weight-based RankBoost (addWRB) given as Algorithm 2. A forward stagewise additive approach [6] is used to search for the strong ranking function F. That is, in each round, a weak ranker f_t is selected and combined with F_{t-1} using coefficient α_t. The source weight is then updated in step 8, where the ensemble function is scaled by λ_t inside the exponent that is further combined with the target weight linearly.

4 Experiments and Results

Evaluation is done on LETOR3.0 benchmark dataset [5] with the Web track documents of TREC 2003 and 2004. We treat each ranking task, namely Home Page Finding (HP), Named Page Finding (NP) and Topic Distillation (TD) [7] as an individual domain. Generally, it is relatively easier to determine a homepage or named page than an entry point of good websites.

We use the same method as [3] to estimate target weights. Note that rank labels are not used for weighting. The *baseline* is a RankBoost directly trained

Algorithm 2. Additive Weight-based RankBoost (addWRB)

1. Initialize $W_1 = \frac{w(x_i,x_j)}{\sum_{i,j} w(x_i,x_j)}$ for all i, j;
2. Set $\lambda_0 = 1$, $F_0 = 0$;
3. **for** $t = 1$; $t \leq T$; $t++$ **do**
4. Select weak ranker $f_t(x)$ using distribution W_t;
5. Set coefficient α_t for $f_t(x)$;
6. $F_t(x) = F_{t-1}(x) + \alpha_t f_t(x)$;
7. Compute λ_t using Eq. 5;
8. For each (x_i, x_j), update source weight using $W_{t+1} = \frac{1}{Z_t} w(x_i, x_j) \exp(\lambda_t (F_t(x_i) - F_t(x_j)))$;
9. **end for**
10. **return** $F(x) = \sum_{k=1}^{T} \alpha_k f_k(x)$

Table 1. MAP results of three adaptation tasks. †, ‡ and ♯ indicate significantly better than baseline, expWRB and linWRB, respectively (95% confidence level).

model	HP→NP		NP→TD			TD→NP	
	Y2003	Y2004	Y2003	Y2004	Y03-Y04	Y2003	Y2004
baseline	0.5834	0.5455	0.1734	0.1657	0.1062	0.4101	0.3061
expWRB	0.5481	0.5206	0.1352	0.1437	$0.1485^{\dagger}$	$0.5493^{\dagger\sharp}$	$0.5159^{\dagger\sharp}$
linWRB	$0.6245^{\dagger\ddagger}$	$0.5824^{\dagger\ddagger}$	$0.1755^{\ddagger}$	0.1444	$0.1433^{\dagger}$	0.3344	0.2239
addWRB	$0.6280^{\dagger\ddagger}$	$0.6025^{\dagger\ddagger}$	$0.2139^{\dagger\ddagger\sharp}$	0.1505	$0.1541^{\dagger}$	$0.5537^{\dagger\sharp}$	$0.5774^{\dagger\sharp}$

on the source domain without target weight. Always we leveraged on decision stumps to implement binary weak rankers.

We examined HP to NP, NP to TD and TD to NP adaptations to study if our algorithms can adapt across similar tasks, from easier to more difficult task and in the reverse case. The MAP results are reported in Table 1.

HP to NP Adaptation. We observe that addWRB outperforms all other algorithms. T-tests indicate that both addWRB and linWRB are significantly better than the baseline and expWRB ($p < 0.02$). This indicates both algorithms can effectively balance the two types of weights.

Note that expWRB failed here. We found that lots of pairs were ordered correctly in HP training, resulting in small source weights. So the target weight inside the exponent quickly dominated the updating of source weight and the same weak ranker was chosen repeatedly. The model becomes not generalizable.

NP to TD Adaptation. NP is rather different from TD. On 2003 data, addWRB works better than other variants, and t-test indicates that the improvements are statistically significant ($p < 0.001$). On 2004 data, all three variants underperform the baseline. This is consistent to [3] using other algorithms due to the shortage of training data from the source domain. Actually, only a half number queries are available in NP04 than NP03. To avoid under-training, we turned to examine NP03 to TD04 adaptation where our algorithms significantly outperformed the baseline ($p < 0.001$).

TD to NP Adaptation. Here we study how our models can adapt from a difficult task to a simple one. We observe that expWRB and addWRB are significantly better than the baseline ($p < 0.0001$) whereas linWRB fails. The target

weight affects linWRB little when source pairs are difficult to rank because of the exponential source weights. So the performance is mainly determined by source weight leading to the failing adaptation. In contrast, expWRB's target weight inside the exponent can effectively balance the growth of source weights.

The Scaling Factor λ. We also examine the influence of λ on 2003 data to unveil how λ reacts to different problem difficulty. We observe that λ in TD to NP is much lower and decreases much faster than NP to TD. Since TD is more complex where lots of pairs are wrong, λ decreases quickly to balance the exponential growth of source weights.

5 Conclusions

We proposed three variants of weight-based boosting models for ranking adaptation based on RankBoost algorithm, namely expWRB, linWRB and addWRB. The challenge is to balance the innate weight distribution of RankBoost and the target weight introduced for adaptation. expWRB and linWRB incorporate target weight in straightforward yet static ways. addWRB uses an additive approach, where the influence of source weight can be scaled dynamically according to the problem difficulty. Experiments demonstrate that the performance of expWRB and linWRB varies with the problem difficulty of source domain, and addWRB consistently and significantly outperforms the baseline.

References

1. Amini, M.R., Truong, T.V., Goutte, C.: A boosting algorithm for learning bipartite ranking functions with partially labeled data. In: Proc. of SIGIR (2008)
2. Freund, Y., Iyer, R., Schapire, R., Singer, Y.: An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research (2004)
3. Gao, W., Cai, P., Wong, K.-F., Zhou, A.: Learning to rank only using training data from related domain. In: Proc. of SIGIR (2010)
4. Liu, T.-Y.: Learning to rank for information retrieval. In: Foundations and Trends in Information Retrieval (2009)
5. Qin, T., Liu, T.-Y., Xu, J., Li, H.: LETOR: A benchmark collection for research on learning to rank for information retrieval. Information Retrieval (2010)
6. Hastie, T., Tibshirani, R., Friedman, J.H.: The elements of statistical learning. Springer, Heidelberg (2001)
7. Voorhees, E.M.: Overview of TREC 2004. In: Proc. of TREC (2004)
8. Xu, J., Li, H.: Adarank: A boosting algorithm for information retrieval. In: Proc. of SIGIR (2007)
9. Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Chen, K., Sun, G.: A general boosting method and its application to learning ranking functions for web search. In: Proc. of NIPS (2007)