# Neural-network Based Regression Model with Prior from Ranking Information

Yajun Qu, Bo Dai and Baogang Hu

*Abstract*— **In this work, a new algorithm, which can incorporate the ranking information as prior knowledge into the regression model, is presented. Comparing with the method that treats the ranking information as hard constraints, We handle ranking reasonably by maximization of Normalized Discount Cumulative Gain (NDCG) as information retrieval (IR) evaluation measure, which is used to evaluate the performance of ranking model. In addition, an upper bound of one minus NDCG is given by weighted pairwise loss, and a connection between weighted pairwise loss and NDCG is also revealed. In this paper, RBF regression model and the pairwise shifted hinge loss and logistic loss are proposed under the suggested approach. One benefit of the proposed approach is that the weighted pairwise loss is more reasonable than the unweighted loss and all the weights are set based on the NDCG. Finally, one synthetic example shows that the method incorporated the ranking as hard constraints into regression model may cause the deteriorated results, but the good performance is shown by the proposed method. Numerical results from three existing benchmark regression problems further confirm the beneficial aspects on the proposed approach.**

## I. Introduction

Incorporating prior knowledge to improve the performance of neural networks is more and more important and is applied to various applications, such as insufficient quantity and quality of the training data [1]. However, the issue that the prior knowledge is incorporated into the model fully and reasonably is still a topic of research, including the discussion of the universal method for one type of particular prior knowledge, such as, monotonicity [1], smoothness [2], equilibrium points [3][4], symmetric [5], and so on. In this paper, a new prior knowledge, called ranking, is discovered and incorporated into the neural networks to improve the regression model.

Ranking is a relationship between a set of categories such that, for any two categories, the first is either 'ranked higher than', 'ranked lower than' or 'ranked equal to' the second, e.g. website are ranked by the relevance of keywords. But this is known as a weaker order, because the ranking only contains the order information of the objects but not the real distance between objects. In fact, in real-world problems, lots of ranking information can be created and available easily. As an example, we consider the task of predicting real estate prices. The users or experts always rate the housing prices by giving the overall characteristics of houses. They can independently give the scoring or ranking of these housing prices roughly according to the features of these houses, such as, area, location, number of bedrooms, etc. For example, if everything else is roughly equal, then the scoring of a 3-bedroom house is higher than a 2-bedroom one. Therefore, our idea is to incorporate this kind of ranking information as prior knowledge into the design of neural networks for improving the performance of the models. In [6], semi-supervised learning with order preferences is proposed to show the improvement, but the ranking information is not considered in it. So in this paper, the algorithm incorporated ranking into the neural networks is proposed.

However, significant differences appear on the numerous approaches for incorporating prior knowledge into the design of neural networks [7][8][9][13]. In [7], four categories of approaches, which are using prior knowledge to prepare training example, initiate the hypothesis, alter objective and augment search, have been classified for incorporating prior knowledge into inductive machine learning. This is referred to as learning with constraints, as addressed by [8]. For better understanding the principles of embedding prior knowledge, the authors in [9] have summarized three basic types for examining the individual approach, such as, structure types [10], algorithm types [11], and data types [12]. From the perspective of explicitness of the models, the three types of embedding principles are ranked in a decedent order. For the maximum utilization of prior knowledge in an explicit means, they proposed a generalized constraint neural network (GCNN) [13] model. However, ranking, as a special prior knowledge, is the central problem for information retrieval (IR) applications. Therefore, different from the above methods, the IR evaluation measure has been used to incorporate the ranking information into the neural network in this paper.

The present paper explores how to add ranking prior knowledge into the conventional neural networks and proposes the algorithm, which reasonably uses the maximization of NDCG as IR evaluation measure, to incorporate the ranking into neural networks. The relationship of NDCG and pairwise loss is given, and an upper bound of one minus NDCG is presented by weighted pairwise loss [14]. In this paper, the pairwise loss, such as hinge loss and logistic loss, will be the main concerned to incorporate the ranking prior knowledge into networks for improving the accuracy and interpretability of the models.

The paper is organized as follows. In the next section we introduce the algorithms of conventional RBF and the loss functions of ranking algorithms briefly. The main contribution of this paper is that a new approach for incorporating ranking prior knowledge into models is presented, and two pairwise losses are used under this approach in Section III.

Yajun Qu, Bo Dai and Baogang Hu are with National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China (email: {yjqu, bdai, bghu}@nlpr.ia.ac.cn).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

In Section IV, experiments on one synthetic example and three benchmark datasets are shown to validate the proposed method. The conclusion and future work are presented in the last section.

## II. Related Formulas

### A. Conventional RBF

RBF networks are used in a variety of applications because they are universal function approximators which can learn unknown functional relationships. Consider a training set $T = \{\mathbf{x}_l, y_l\}_{l=1,2,\cdots,N}$, where $\mathbf{x}_l \in \mathbf{R}^{1\times n}$ is an input vector, $y_l \in \mathbf{R}$ denotes the vector of desired network outputs for the input $\mathbf{x}_l$ and $N$ is the number of training set. The output of the RBF network is calculated according to

$$f(\mathbf{x}) = \mathbf{\Phi}(\mathbf{x})\mathbf{W} \tag{1}$$

where $f(\cdot)$ is the output, $\mathbf{W} = [\omega_0, \omega_1, \ldots, \omega_h]^T$ are the weights of the network, and $\mathbf{\Phi}(\mathbf{x}) = [1, \phi_1(\mathbf{x}), \ldots, \phi_h(\mathbf{x})]$, $h$ is the number of neurons in the hidden layer, and $\phi_i(\cdot)$ is a basis function of the RBF network. In this paper, the Gaussian RBF: $\phi_i(\mathbf{x}) = exp(-\|\mathbf{x} - \mathbf{c}_i\|^2/\sigma_i^2)$, $i = 1, 2, \ldots, h$ is discussed, where $\mathbf{c}_i \in \mathbf{R}^{1\times n}$ are the RBF centers and parameter $\sigma_i$ controls the "width" of the RBF. The most common approach is to determine the centers and widths of the radial basis functions at first, then the weights are estimated subsequently. Some effective methods [15][16] and simple heuristic relationship [17] have been introduced to determine the centers and the widths, respectively.

Once the centers and widths are chosen, the network training task is changed to determine the appropriate settings of the weights between the neurons. Therefore, the optimal set of weights minimizes the performances measure

$$\min_{\mathbf{W}\in\mathbf{R}^{h+1}} : E(\mathbf{W}) = \frac{1}{2}(\mathbf{y} - \mathbf{\Phi}\mathbf{W})^T(\mathbf{y} - \mathbf{\Phi}\mathbf{W}) \tag{2}$$

where $\mathbf{y} = [y_1, \cdots, y_N]^T \in \mathbf{R}^{N\times 1}$ denotes the component of vector of desired network outputs and $\mathbf{\Phi} = [\mathbf{\Phi}^T(\mathbf{x}_1), \mathbf{\Phi}^T(\mathbf{x}_2), \ldots, \mathbf{\Phi}^T(\mathbf{x}_N)]^T \in \mathbf{R}^{N\times(h+1)}$(the first row is 1). In order to avoid over-fitting, we control the complexity of the neural network model by the addition of a regularization term to the error function. The simplest regularizer is the quadratic, giving a regularized error of the form

$$\min_{\mathbf{W}\in\mathbf{R}^{h+1}} : E(\mathbf{W}) = \frac{1}{2}(\mathbf{y} - \mathbf{\Phi}\mathbf{W})^T(\mathbf{y} - \mathbf{\Phi}\mathbf{W}) + \frac{\lambda}{2}\mathbf{W}^T\mathbf{W} \tag{3}$$

where $\lambda$ is the regularization coefficient, which can determine the model complexity. Gradient descent methods or linear least squares strategies can be used to determine the weights of the Eq. (3). Due to the speed and accuracy of the training, linear least squares algorithm is used in this paper. Minimizing the performance measure $E(\mathbf{W})$ is achieved by

$$\frac{\partial E(\mathbf{W})}{\partial \mathbf{W}} = -\mathbf{\Phi}^T(\mathbf{y} - \mathbf{\Phi}\mathbf{W}) + \lambda\mathbf{W} = \mathbf{0} \tag{4}$$

Solving for $\mathbf{W}$, we have

$$\mathbf{W} = [\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I}]^{-1}\mathbf{\Phi}^T\mathbf{y} \tag{5}$$

where $\mathbf{I}$ is the identity matrix. From (5) the weights of RBF network are determined and the problem of network training can have a closed-form solution.

### B. Learning to Rank

Learning to rank is aimed at creating the ranking model using the training data and machine learning techniques. Many methods have been proposed to address the problem of learning to rank [18][19] and applied to information retrieval (IR) [18][20]. At first, we give some notations. Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$ be the input space of the objects set, which we are interested in ranking, $\mathbf{Y} = \{y_1, \ldots, y_p\}$ be the output space representing the levels of relevance of objects as the ground truth list. Let $\mathbf{F}$ be the function class, $f \in \mathbf{F}$, and let $y(i)$ be the index of object which is ranked at position $i$. The task is to learn a ranking function from the training data by minimization of a loss function. The pairwise algorithms and listwise algorithms are two state-of-the-art approaches.

In the pairwise approach, the problem of learning-to-rank is approximated by a binary classification problem, which is mainly to find which object is better in a given pair of objects. The goal of this approach is to minimize average number of inversions in ranking. The loss function is defined as:

$$L(f; \mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{p-1} \sum_{j=1, y(i)<y(j)}^{p} l(f(\mathbf{x}_i) - f(\mathbf{x}_j)) \tag{6}$$

where different functions $l(\cdot)$ are used in different pairwise ranking algorithms. For example, Ranking SVM [19], hinge loss $l(x) = max(1 - x, 0)$; and in RankNet [18], logistic loss $l(x) = log(1 + e^{-x})$. In the listwise approach, it directly takes ranked lists of objects as instances and trains a ranking function through the minimization of a listwise loss function defined on the predicted list and the ground truth list. The ListMLE [21] is one example of listwise approach. In this paper, the loss function of pairwise approach will be mainly concerned to handle the ranking.

In addition, in order to measure the performance of a ranking model, IR evaluation measures, such as Mean Average Precision (MAP) and Normalized Discount Cumulative Gain (NDCG), are widely used [22][23]. MAP can only handle cases with binary judgment: "relevant" or "irrelevant", while NDCG can handle multiple levels of relevant judgment. Therefore, we give the definition of NDCG value of a ranking list as follow:

$$NDCG(\pi, R) = \frac{1}{N_p}\sum_{j=1}^{p}\frac{2^{r_j} - 1}{log(1 + j)}, \tag{7}$$

where $r_j$ is the relevance of the $jth$ object in the ranking list $\pi$ and $R$ is their ground truth permutation, and $N_p$ is a normalizer to ensure $NDCG \in [0, 1]$. In a perfect ranking algorithm, NDCG is 1.0.

## III. RBF Networks with Ranking Prior

As seen in previous section, the training data with target value are used to learn unknown function relationships by

conventional RBF network, and the training data with relevance label of object are used to create the ranking model by approaches of learn-to-rank. In this section, a new approach, which is aimed at incorporating the ranking information into the network, is proposed.

At first, we give some notations. Consider a training set $T = (\mathbf{x}_l, y_l)_{l=1,2,\cdots,N}$, where $y_l$ denotes the target value of the input vector $\mathbf{x}_l$ and $N$ is the number of training set. The ranking data $P = (\mathbf{x}_j, r_j)_{j=1,2,\cdots,p}$, where $r_j \in \{1, 2, \ldots, K\}$ is the relevance label of input vector $\mathbf{x_j}$ and $p$ is the number of ranking data. The ranking data satisfy the following condition: $f(\mathbf{x}_i) > f(\mathbf{x}_j)$, if $r_i < r_j, i, j \in P$, where $f(\cdot)$ is defined by Eq. (1), which is the key point, because the function $f(\cdot)$ is defined not only our regression model but also the ranking scoring function. That is to say, it can connect the ranking to the regression model in regression. For simplicity, we assume that $r_i \neq r_j, \forall i \neq j, i, j \in P$. Because the ranking data only contain the ranking information rather than the target value of the data, it is weaker. But we would like to use the ranking data to improve the regression model. However, the pairwise algorithms and the listwise algorithms are the main approaches to deal with the ranking information in learn-to-rank. In this paper, the pairwise approaches are used to handle the ranking information. That is to say, the ranking information has been divided into pairs of ranking $[f(\mathbf{x}_i), f(\mathbf{x}_j)], r_i < r_j, i, j \in P$, which means the scoring function $f(\mathbf{x}_i)$ is larger than $f(\mathbf{x}_j)$.

According to the summary in [9], there are many types of schemes, such as generalized constraint neural network (GCNN), regularization theory and virtual examples, all of which can embed ranking as prior knowledge into systems. A simple and natural way is to treat the pairs of ranking as constraints to embed the regression model. Therefore, the pairs of ranking can be incorporated into the RBF (3), by adding the ranking as constraints to it as follows

$$\begin{array}{ll} \min\limits_{\mathbf{W}}: & J(\mathbf{W}) = E(\mathbf{W}; T) \\ s.t. & y_{si}(\mathbf{\Phi}(\mathbf{x}_s)\mathbf{W} - \mathbf{\Phi}(\mathbf{x}_i)\mathbf{W}) \geqslant 0, \\ & p \geqslant i > s \geqslant 1. \end{array} \tag{8}$$

where $y_{si}$ denotes $sign[r_i - r_s]$, which is used to satisfy the condition of ranking. This linear constrained quadratic programming will ensure that each of the pairs of ranking can maintain their original position. However, the pair of ranking is the order relationship rather than a hard constraint. So the above way to add the ranking into the model misunderstand the meaning of ranking, much less use the ranking information fully and reasonably. We would like to use a reasonable manner to handle the ranking information. Due to ranking is our main concerned, the NDCG is the criterion to measure the performance of a ranking model. Therefore, the NDCG criterion is a good choice to incorporate ranking into regression model in this paper. In this section, we will utilize the NDCG criterion to derive the algorithm that can incorporate the ranking into the RBF network reasonably.

We add the NDCG criterion as regularization to RBF model. But in ranking the maximization of DNCG is used to optimize for a good ranking model. Because the $NDCG \in [0, 1]$, the minimization of $(1 - NDCG)$ is added to the RBF model. Therefore, according to the standard error function of RBF network, we have

$$\min_{\mathbf{W}}: J(\mathbf{W}) = E(\mathbf{W}; T) + \mu(1 - NDCG(\mathbf{W}; P)) \tag{9}$$

where $E(\mathbf{W}; T)$ is an error function defined by Eq. (3), $\mu$ is a weight of regularization and $NDCG(\mathbf{W}; P)$ is defined by Eq. (7). The objective function, which can deal with the ranking reasonably, is given by above formula, but such problem is still hard to optimize, because the NDCG is dependent on the ranking position of objects induced by the ranking function, not the numerical values output by the ranking function. Therefore, we will do further analysis on NDCG and hope to obtain an equivalent form optimized easily from the above problem.

Let ranking data set be $P$ and their ground truth permutation be $R$. The ranking is divided into several sub-ranking lists, and for each sub-ranking $s$, let $\mathbf{x}_{R(s)}$ be the object ranked at $s$ position in $R$ and $\pi(\mathbf{x}_{R(s)})$ stand for the rank position of the object $\mathbf{x}_{R(s)}$ by $\pi$, which is an instance of ranking, and other objects, which are denoted as $P_s = \{x_{R(s)}, \ldots, x_{R(p)}\}$, are all ranked below $s$ position in $R$. Therefore, at each step $s$, we have the loss

$$\begin{aligned} l_s(f; P_s, R) &= Z_s \sum_{i=s+1}^{p} l(f(\mathbf{x}_{R(s)}) - f(\mathbf{x}_{R(i)})) \\ &= 1 - Z_s \sum_{i=s+1}^{p} l(f(\mathbf{x}_{R(i)}) - f(\mathbf{x}_{R(s)})) \\ &= 1 - \bar{l}_s \end{aligned} \tag{10}$$

where $Z_s$ is a normalizer to ensure $l_s, \bar{l}_s \in [0, 1]$. If $l_s = 0$, that is $\bar{l}_s = 1$, then $\pi(\mathbf{x}_{R(s)}) < \pi(\mathbf{x}_{R(i)})$ for $\forall i > s$.

According to the definition of NDCG in (7), the loss function in (6) and Eq. (10), we have the weighted pairwise loss function

$$L(\pi, R) = \sum_{s=1}^{p-1} \alpha(s) l_s \tag{11}$$

where $\alpha(s)$ is the weight to stand for the importance of the loss $l_s$ at step $s$, and if $\alpha(s) = 1/Z_s, s = 1, \cdots, p-1$, then the Eq. (11) is the general ranking loss function (6).

*Theorem 3.1:* If $\alpha(s) = \dfrac{2^{R_s} - 1}{N_p log(R(s) + 1)}$, then we have

$$(1 - NDCG) \leqslant L(\pi, R).$$

*Proof:* According to the Eq. (7), we have

$$\begin{aligned} & 1 - NDCG \\ &= \frac{1}{N_p} \sum_{s=1}^{p} [\frac{2^{R_s} - 1}{log(s+1)} - \frac{2^{\pi_s} - 1}{log(s+1)}] \\ &= \sum_{s=1}^{p} \frac{2^{R_s} - 1}{N_p} [\frac{1}{log(s+1)} - \frac{1}{log(\pi(\mathbf{x}_{R(s)}) + 1)}] \end{aligned}$$

Because the position of object $\mathbf{x}_{R(p)}$ at permutation $\pi$ must be no more than $p$, that is $\pi(\mathbf{x}_{R(p)}) \leqslant p$, then $\dfrac{1}{log(p+1)} \leqslant$

$\frac{1}{log(\pi(\mathbf{x}_{R(p)})+1)}$. Therefore, we can get

$$\begin{aligned} & 1-NDCG \\ & \leqslant \sum_{s=1}^{p-1} \frac{2^{R_s}-1}{N_p log(s+1)}[1-\frac{log(s+1)}{log(\pi(\mathbf{x}_{R(s)})+1)}] \end{aligned}$$

Due to $\bar{l}_s \leqslant 1$, and we can obtain that

$$1-NDCG \leqslant \sum_{s=1}^{p-1} \alpha(s)[1-\frac{log(s+1)}{log(\pi(\mathbf{x}_{R(s)})+1)}\bar{l}_s]$$

where $\alpha(s) = \frac{2^{R_s}-1}{N_p log(s+1)}$. In addition, according to the Eq. (10), for each step $s$, if $\bar{l}_s = 1$, then $f(\mathbf{x}_{R(s)}) > f(\mathbf{x}_{R(i)})$, that is $\pi(\mathbf{x}_{R(s)}) < \pi(\mathbf{x}_{R(i)}))$ for $\forall i > s$. In other words, at each step $s$, all objects, which are all ranked below $s$ position in $R$, are still ranked below the object $\mathbf{x}_{R(s)}$ in the ranking $\pi$. Therefore, we have $\pi(\mathbf{x}_{R(s)}) \leqslant s$, and we can get $log(\pi(\mathbf{x}_{R(s)})+1) \leqslant log(s+1)$. So we can obtain that

$$1-NDCG \leqslant \sum_{s=1}^{p-1} \alpha(s)[1-\bar{l}_s] = \sum_{s=1}^{p-1} \alpha(s) l_s$$

■

The above theorem not only gives the direction to solve the problem (9), but also provides the connection between the pairwise algorithms loss function and the NDCG. In other words, the minimization of $(1-NDCG)$ can result in the minimization of the weighted pairwise algorithms loss, and the weighted pairwise loss is an upper bound of $(1-NDCG)$. The similar result of above has also been given in [14].

According to the Theorem 3.1, the loss of (9) is rewritten as

$$\min_{\mathbf{W}} : J(\mathbf{W}) = E(\mathbf{W};T) + \mu L(\mathbf{W};P) \tag{12}$$

where $L(\mathbf{W};P)$ is defined by Eq. (11). Up to now, the task that the ranking is incorporated into the RBF network has been completed basically. Additionally an upper bound of the loss function (9) is also given by the above equation. In fact, the Eq. (12) provides us a new approach to incorporate the ranking as prior knowledge into the networks. Margin-based convex surrogate loss, such as hinge loss, exponential loss and logistic loss, can be used to optimize the above convex loss. In this paper, the shifted hinge loss and logistic loss will be used to express our approach, which can also be used by other loss functions.

### *A. Pairwise Hinge Loss*

In this section, the shifted hinge loss function is used to the proposed approach. Once the loss function is selected, the specific form of the $L(\mathbf{W};P)$ in (12) can be given as follows

$$L(\mathbf{W};P) = \sum_{s=1}^{p-1} \alpha(s) Z_s \sum_{i=s+1}^{p} [1-C(f(\mathbf{x}_{R(s)}) - f(\mathbf{x}_{R(i)}))]_+ \tag{13}$$

where $\alpha(s)$ is defined by the Theorem 3.1, $C$ is a normalizer to ensure $(f(\mathbf{x}_{R(s)}) - f(\mathbf{x}_{R(i)})) \in [-1,1]$ and $[x]_+ = max(x,0)$. Note that the shifted hinge loss we used is $[1-Cx]_+$, which is different from the pairwise hinge loss $[1-x]_+$. The main reason is that in pairwise ranking model the predictive function $f$ is denoted as the classifier, which do not care about the real values of the objects but their signs, while in our regression model the target values are our mainly concerned. So on the basis of definition of hinge loss, if $f$ satisfies all the ranking data, $L(\mathbf{W};P)$ should be zero; and if $f$ violates some, the loss will increase. So we give the normalizer $C$ to keep the rules.

We transform (13) into a quadratic program by adding auxiliary variables $\epsilon_{si}$ and put it into the Eq. (12). Therefore, the quadratic program for incorporating ranking into RBF regression model is

$$\begin{aligned} \min_{\mathbf{W},\epsilon_{si}} : \quad & J_H(\mathbf{W}) = E(\mathbf{W};T) + \mu \sum_{s=1}^{p-1} \beta(s) \sum_{i=s+1}^{p} \epsilon_{si} \\ s.t. \quad & \mathbf{\Phi}(\mathbf{x}_{R(s)})\mathbf{W} - \mathbf{\Phi}(\mathbf{x}_{R(i)})\mathbf{W} \geqslant \frac{1}{C} - \epsilon_{si} \\ & p \geqslant i > s \geqslant 1, \\ & \epsilon_{si} \geqslant 0. \end{aligned} \tag{14}$$

where $\beta(s) = C\alpha(s)Z_s$. From the above equation we can see that the auxiliary variables are weighted, which is more reasonable than the unweighted variables. In addition, the weights can be determined easily by the NDCG, which is also more reasonable than them set by randomly or empirically. Another feature, which is different from other algorithms, is the constant coefficient $1/C$ rather than zero. For example, consider an order relationship between two instances as $f(x_i) > f(x_j)$. In general algorithms, this order is added to those approaches as constraint $f(x_i) - f(x_j) \geqslant 0$. But it is an order relationship rather than a hard constraint. Therefore, it is more reasonable that constant coefficient $1/C$ is added to soft the constraint.

In order to optimize the problem (14) conveniently, according to the conditions of Karush-Kuhn-Tucker (KKT), the dual problem can be obtained and written in the form of vector as follows

$$\begin{aligned} \max_{\mathbf{\Theta}} : \quad & J_{DH}(\mathbf{\Theta}) = \mathbf{Z}\mathbf{\Psi}^T\mathbf{\Theta} - \frac{1}{2}\mathbf{\Theta}\mathbf{\Psi}\mathbf{H}^{-1}\mathbf{\Psi}^T\mathbf{\Theta} \\ s.t. \quad & \mu\beta(s) \geqslant \theta_{si} \geqslant 0, p \geqslant i > s \geqslant 1. \end{aligned} \tag{15}$$

where $\mathbf{\Theta} = [\theta_{12}, \ldots, \theta_{1p}, \theta_{21}, \ldots, \theta_{(p-1)p}]^T$ is the vector of Lagrange multipliers, $\mathbf{H} = \mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{I}$, $\mathbf{Z} = \frac{1}{C}\mathbf{I} - \mathbf{y}^T\mathbf{\Phi}\mathbf{H}^{-1}$, and $\mathbf{\Psi} = [(\mathbf{\Phi}(\mathbf{x}_{R(1)}) - \mathbf{\Phi}(\mathbf{x}_{R(2)}))^T, \ldots, (\mathbf{\Phi}(\mathbf{x}_{R(s)}) - \mathbf{\Phi}(\mathbf{x}_{R(i)}))^T, \ldots, (\mathbf{\Phi}(\mathbf{x}_{R(p-1)}) - \mathbf{\Phi}(\mathbf{x}_{R(p)}))^T]^T, p \geqslant i > s \geqslant 1$. Therefore, the linear constrained RBF network training algorithm (Algorithm 1) in [4] can be used to solve the above dual problem. And $j_{max}$ can be calculated by

$$j_{max} = \arg \min_{j \in I \setminus S_k, \theta_j < 0} \{J_{DH}(\mathbf{\Theta}^{(k)}), S_k \leftarrow j \cup S_k\} \tag{16}$$

Finally, the problem (14) can be solved and the regression model can be obtained.

**Algorithm 1** Linear Constrained RBF Network Training

1: given active set $S_1 \leftarrow \emptyset$, $k \leftarrow 1$;
2: obtained $\mathbf{\Theta^{(k)}}$ from

$$\begin{aligned} \max_{\mathbf{\Theta}^{(k)}}: \quad & J_{DH}(\mathbf{\Theta}^{(k)}) \\ s.t. \quad & \theta_j^{(k)} = 0, j \in S_k \end{aligned}$$

a. if $\exists j \in I \setminus S_k$ and $\theta_j < 0$, then jump 3;
b. else if Lagrange multipliers of dual problem $\upsilon_j^{(k)} \geqslant 0, \forall j \in S_k \cap I$, then the algorithm stop;
c. else calculate $j_k$ by solving

$$\upsilon_{j_k}^{(k)} = \min_{j \in S_k \cap I} \upsilon_j^{(k)} < 0,$$

$S_k \leftarrow S_k \setminus \{j_k\}$, jump 4;

3: find $j \in I \setminus S_k$ and $\theta_j < 0$, then $S_k \leftarrow S_k \cup \{j_{max}\}$, jump 2;
4: $S_{k+1} \leftarrow S_k$; $k \leftarrow k+1$; jump 2.

### B. Pairwise Logistic Loss

In this section, another pairwise loss function, called logistic loss, is used in the suggested method to explain further that our proposed approach is an universal approach rather than a special case. According to logistic loss function, the specific form of the $L(\mathbf{W}; P)$ in (12) can be given as follows

$$L(\mathbf{W}; P) = \sum_{s=1}^{p-1} \beta(s) \sum_{i=s+1}^{p} log(1 + e^{-(f(\mathbf{x}_{R(s)}) - f(\mathbf{x}_{R(i)}))}) \tag{17}$$

where $\beta(s) = \alpha(s) Z_s$. Putting the above form into (12) directly, the final form, which incorporates the ranking information into the RBF regression model with logistic loss, can be obtained as $J_L(\mathbf{W})$. Because the logistic loss function is continuous and differentiable, the optimization algorithm adjusts the weights iteratively based on the loss gradient

$$\mathbf{W}^{k+1} = \mathbf{W}^k - \eta \frac{\partial J_L(\mathbf{W})}{\partial \mathbf{W}}|_{(\mathbf{W}=\mathbf{W}^k)} \tag{18}$$

where $k$ is the epoch index, $\eta$ is the learning rate, and the loss gradient $\partial J_L(\mathbf{W})/\partial \mathbf{W}$ is

$$\frac{\partial J_L(\mathbf{W})}{\partial \mathbf{W}} = \frac{\partial E(\mathbf{W})}{\partial \mathbf{W}} + \frac{\partial L(\mathbf{W})}{\partial \mathbf{W}} \tag{19}$$

where $\partial E(\mathbf{W})/\partial \mathbf{W}$ is defined by (4) and $\partial L(\mathbf{W})/\partial \mathbf{W}$ is

$$\frac{\partial L(\mathbf{W})}{\partial \mathbf{W}} = \sum_{s=1}^{p-1} \beta(s) \sum_{i=s+1}^{p} \frac{-t_{si}(\mathbf{\Phi}(\mathbf{x}_{R(s)}) - \mathbf{\Phi}(\mathbf{x}_{R(i)}))^T}{1 + t_{si}}$$

where $t_{si} = e^{-(f(\mathbf{x}_{R(s)}) - f(\mathbf{x}_{R(i)})}$. Therefore, the gradient descent algorithm (Algorithm 2) can be used to adjust the weights. So the regression model can also been obtained by above algorithm.

The above mentioned methods are all used in the RBF regression model. In fact, other nonlinear regression models can also be used in the proposed approach (12), such as general neural networks regression model, SVR and so on.

**Algorithm 2** Gradient Descent Algorithm

1: given $\varepsilon$, $\eta$, $\mathbf{W}^1, k \leftarrow 1$;
2: compute $\nabla J_L^k = \frac{\partial J_L(\mathbf{W})}{\partial \mathbf{W}}|_{(\mathbf{W}=\mathbf{W}^k)}$ by (19), if $\nabla J_L^k \leqslant \varepsilon$, then the algorithm stop;
3: update the weights by (18);
4: $k \leftarrow k+1$, jump 2.

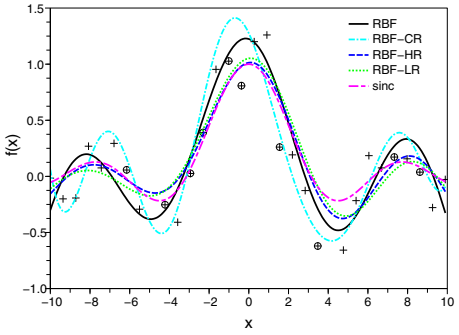



Fig. 1. The results of RBF, RBF-CR, RBF-HR and RBF-LR in example A. 10 points generated from $sinc$ function added Gaussian noise $N(0, 0.2^2)$, and is ranked based on their values. But this ranking is used as prior without the true target values. $+$ and $\oplus$ stand for the training data and ranking data, respectively.

In addition, more than one ranking list can also be handled by this approach, and the only work we do is to calculate all losses $L(\mathbf{W}; P^l)|_{l=1,2,...,M}$ of the ranking lists, where each $P^l$ stands for one ranking list data, and put all of them together as the total loss. Therefore, in this paper, we only take RBF network and one ranking list as an example to propose the new approach, which can incorporate ranking list reasonably. Other regression models and several ranking lists used in the suggested approach are only the extension and applications.

## IV. Simulations and Considerations

In this section, an extensive experimental study of the proposed method on artificial and real-world data sets is presented. The first one is synthetic example that is performed to demonstrate the effectiveness of the proposed method; the others show the benefit of the method on three real-world data sets. For each example, we used 10-fold cross validation to find the optimal size of RBF network $h$, and RBF centers and widths were selected by kcenter and simple heuristic relationship [17], respectively. We used the acronym RBF (conventional RBF) for (3), RBF-CR(RBF with ranking as constraints) for (8), RBF-HR (RBF with ranking as hinge loss) for (12) with hinge loss and RBF-LR (RBF with ranking as logistic loss) for (12) with logistic loss. The mean squared error (MSE) is defined below

$$\mathbf{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \widehat{y}_i)^2 \tag{20}$$

TABLE I

COMPARISON OF PERFORMANCE FOR RBF, RBF-CR, RBF-HR AND EBF-LR IN EXAMPLE A. ($N_{test} = 200$, AND NOISE $N(0, 0.2^2)$)

| Model | $p$ | $N_{train}$ | Total Num. of Free Para. | MSE(Mean/Var.) |
|---|---|---|---|---|
| RBF | 0 | 20 | 8 | 0.0276/0.0188 |
| RBF-CR | 10 | 20 | 10 | 0.0342/0.0329 |
| RBF-HR | 10 | 20 | 9 | 0.0148/0.00714 |
| RBF-LR | 10 | 20 | 9 | 0.0212/0.0110 |

where $y_i$ and $\widehat{y}_i$ represent the observations of network output and the model predictions, respectively.

### A. A synthetic example

In this section, we use a synthetic example to illustrate the effectiveness of the ranking information as prior knowledge. To get data from [24], $sinc$ is given as follows

$$sinc = \frac{sin(x)}{x}, x \in [-10, 10]. \tag{21}$$

20 training data were generated uniformly from the function (21) added with Gaussian noise of zero mean and standard deviation 0.2, and 200 testing data were also generated from the same function without any noises. Due to the lack of the training data and noise, conventional RBF network could not work well. In Fig. 1, we can see that the outputs of the conventional RBF network (solid black line) are very different from the target function's (dot dash red line).

In order to show the improvement of our method, we randomly selected 10 points generated from $sinc$ function added with the same Gaussian noise $N(0, 0.2^2)$, ranked these points based on their values, and used this ranking information as the prior knowledge without the true target values. We did not know the actual target values of these points, but we would like to use the ranking to improve the regression model. Therefore, the proposed approach (12) was used to deal with this problem, and hinge loss and logistic loss were used to validate the new approach. For this experiment we used 10-fold cross validation to find the optimal weights $\lambda$ and $\mu$.

Fig. 1 shows the outputs of RBF network with and without ranking data. At around points of $x = -5$, the function approximation of conventional RBF, caused by noise and lack of training data, can not work well and RBF-CR also can not deal with this problem well, while the approximation can be improved in the proposed model by the ranking. From Table I the proposed methods with ranking can be improved indeed, but RBF-CR causes a larger testing MSE, which verify that incorporating ranking into the model by using hard constraints is unreasonable. Because the ranking information is only the order of the target values in these points, rather than the true distance among their targets or hard constraints. In fact, the main reason is that ranking data have noise, and the noise is imposed to the model by treating ranking as hard constraints. In addition, Due to the ranking information just contains the order of these points, the ranking as prior knowledge is weaker, and the improvement by this prior is limited. But we would like to know what influence it has on the regression model if we change the number of training data; what if we change the number of ranking data; and what if we change the standard deviation of noise?

All experiments were repeated for 20 random trials, and different approaches shared the same data, including $N_{train}$ training data, $p$ ranking data and 200 test data. Fig. 2 addresses the above three questions. With the increasing of the number of training data, the improvements of the proposed approaches decrease in Fig. 2 (a). Because of the increasing of training data, the model has a good performance, and the benefit from the ranking diminishes. Another reason may be that increasing of training data leads to more weaker ranking data. Anyway, the proposed approaches are useful when training data are scarce. Many weak prior knowledge is likely to be a strong prior. In Fig. 2 (b) shows that the benefit is getting better as the weak ranking grows. In Fig. 2 (c), as the noise increases, the proposed approaches compared with conventional RBF still have good performance. In addition, in Fig. 2, we can see that RBF-CR causes a larger testing MSE in most cases, which further illustrates that hard constraints misunderstand the ranking information and RBF-CR can not improve the model with ranking. However, the suggested methods can achieve it. Therefore, according to this example, the proposed method can improve the approximation. When the number of ranking increase or training data are scarce, the improvement will be better.

TABLE III

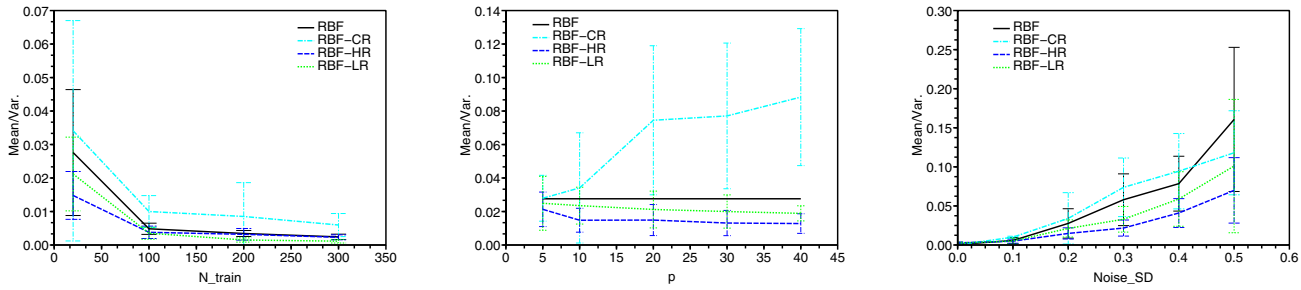
CHARACTERISTICS OF DATA SETS

| Data Set Name | $N$ | $d$ | Target Feature |
|---|---|---|---|
| Pollution | 60 | 16 | Mortality rate |
| Boston Housing | 506 | 14 | MEDV |
| California Housing | 20640 | 9 | House price |

### B. Benchmark datasets

In this section, the results from a comparative study of RBF-HR, RBF-LR and conventional RBF algorithms using real-world data sets (Available at http://funapp.cs.bilkent.edu.tr/DataSets/) are reported (Because RBF-CR is not a good method to incorporate ranking into regression model, it does not attend in this example). Table III gives the characteristics of the data sets, where $N$ is the available number of observations and $d$ denotes the number of attributes, which means the dimensions of the observations. In the last column, the names of target features are presented.

In fact, prior knowledge should be prepared and obtained by expert system in advance. However, in this paper, in order to demonstrate the effectiveness of the suggested method, $p$ observations, which are one part of data set and are not used in training data or testing data, will be used to generate the simulated ranking information. So the whole data set will be divided into three parts: $p$ prior observations, $N_{train}$ training data and $N_{test}$ testing data. Note that the true values of $p$




(a) (b) (c)

Fig. 2. Example A: A synthetic example. The influences from different $N_{train}$, $p$ and standard deviation of noise $\sigma_{noisy}$ on the performance of model are shown in this Fig. When $p = 10$ and $\sigma_{noisy} = 0.2$ are fixed, different effects vary from different $N_{train}$ in (a). If $N_{train} = 20$ and $\sigma_{noisy} = 0.2$ are fixed, the improvement is affected by the changing of $p$ in (b). When $p = 10$ and $N_{train} = 20$, (c) shows the effects from the different $\sigma_{noisy}$.

TABLE II

COMPARISON OF MODELING PERFORMANCE FOR RBF, RBF-HR AND RBF-LR IN EXAMPLE B. THE ERROR MEANS AND STANDARD DEVIATION OF THE MSE ON THE 20 GROUPS OF TEST DATA.

| Data Set | $p$ | $N_{train}$ | $N_{test}$ | MSE(Mean/Var.) RBF | MSE(Mean/Var.) RBF-HR | MSE(Mean/Var.) RBF-LR | RBF-HR Impro. | RBF-LR Impro. |
|---|---|---|---|---|---|---|---|---|
| Pollution | 10 | 10 | 40 | 0.0672/0.0326 | 0.0411/0.0118 | 0.0466/0.0152 | 38.8% | 30.6% |
| Boston Housing | 20 | 20 | 466 | 0.0496/0.0138 | 0.0390/0.00379 | 0.0421/0.00925 | 21.4% | 15.1% |
| California Housing | 20 | 20 | 20600 | 0.0818/0.0258 | 0.0576/0.00534 | 0.0605/0.00874 | 29.6% | 26.0% |

prior observations are never given out.

Each data set is randomly divided into three collections: $p$ prior observations, $N_{train}$ training data and $N_{test}$ testing data. The specific values of the parameters in different data sets are shown in Table II. A standard RBF is trained on these data without any prior, while our model with the ranking information. For contrast, the target feature of each data set was normalized in $[0, 1]$, which was also handled in the similar way in [25]. We used 10-fold cross validation to find the optimal weights $\lambda$ and $\mu$. The parameters were tuned on a $10 \times 10$ logarithmic grid in $10^{-2} \leqslant \lambda \leqslant 10^{2}$ and $10^{-3} \leqslant \mu \leqslant 10^{3}$. The programming ran 20 times for getting 20 groups of test MSE. Table II shows the average and standard deviation of the 20 test MSE. From the table, we can see that the proposed model not only can decrease the average of test MSE, which means the accuracy has been improved, but also can decrease the standard deviation of the test MSE, which means the generalization capability has been improved. Therefore, the suggested method can achieve high performance when ranking information is used for learning.

In order to validate the problems raised in Example A, for each data set, we did the similar experiments. Fig. 3 shows the experimental results for each data set. The results are consistent with Example A's. With the increasing of training data, the improvement of the proposed approach decreases. The benefit from the ranking is getting better as the ranking data grow. In addition, from these three data sets, we can find that if the testing MSE is large, which means the training data have too much noise, the improvements of proposed methods are more obvious, such as California data; if the training data are scarce, the benefit from ranking is better, such as Pollution data; otherwise the improvements are general, such as Boston data. Therefore, the proposed methods are very useful when training data are scarce or the training data have large noise.

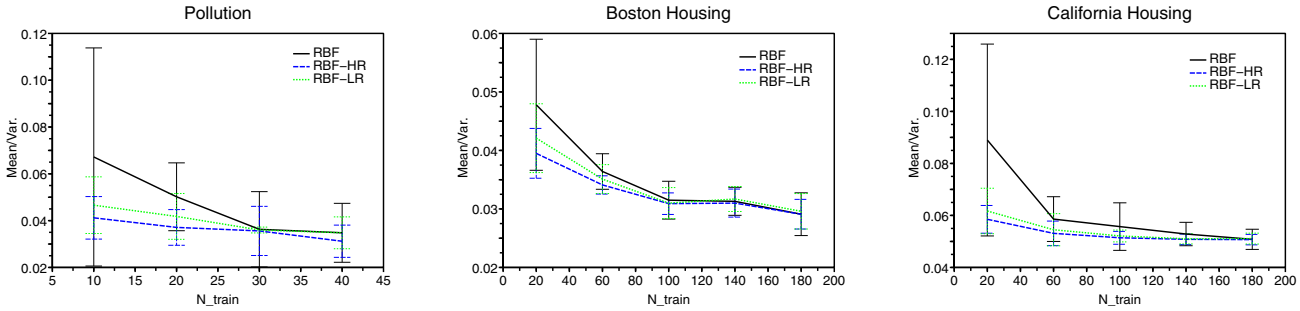
## V. CONCLUSIONS AND FUTURE WORKS

In this paper, we present a new approach, which incorporates the ranking information into the regression model. In the process of handling ranking, the relationship between NDCG and pairwise loss is given, and an upper bound of one minus NDCG is presented. In addition, the RBF model and pairwise shifted hinge loss and logistic loss are used under the proposed approach in this paper. Comparing with the conventional regression model, the proposed methods can improve the performance of the model, and the benefit is getting better, when the training data are scarce or have large noise, or the ranking data grow. One synthetic example and three benchmark regression problems demonstrated it.

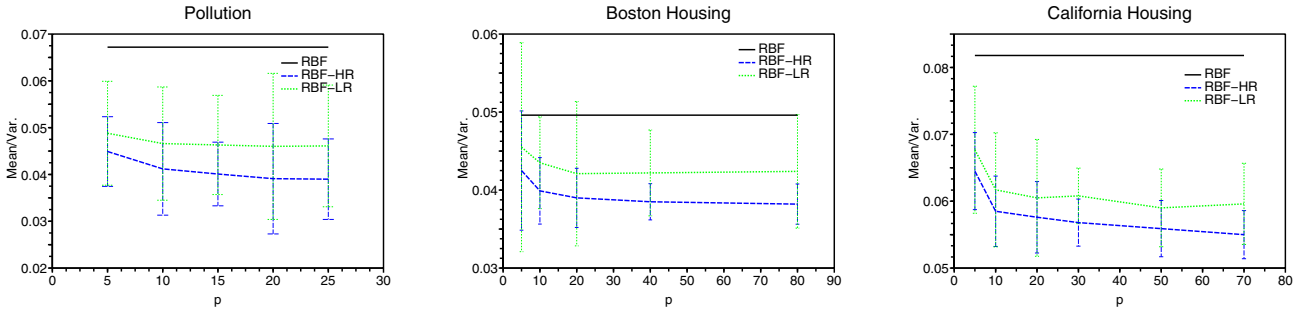
Because pairwise losses are used in this paper, the listwise, which can directly handle the ranking list of objects as instances, will be expanded into the proposed approach for the future work.

## ACKNOWLEDGMENT

This work was supported in part by NSF of China (No. 60275025) and MOST of China (No. 2007DFC10740).




(a) MSE - $N_{train}$




(b) MSE - $p$

Fig. 3. Example B: Benchmark example. The influences from different $N_{train}$ and $p$ on the performance of proposed model are shown in this Fig. The values of $N_{train}$ and $p$ are from the Table II. When one is fixed, the benefit of the ranking varies from the changing of the other one.

## REFERENCES


[1] A. Minin, B. Lang, and M. Velikova, "Comparison of universal approximators incorporating partial monotonicity by structure, " *Neural Networks*, 2009.

[2] S. Haykin, *Neural Networks, second ed.*, A Comprehensive Foundation, Prentice-Hall, New York, 1999.

[3] F. Lauer and G. Bloch, "Incorporating prior knowledge in support vector regression," *Machine Learning*, vol. 70, pp. 89-118, 2008.

[4] Y. J. Qu and B. G. Hu "RBF networks for nonlinear models subject to linear constraints," *2009 IEEE International Conference on Granular Computing, GrC 2009*, pp. 482-487, 2009.

[5] S. Chen, A. Wolfgang, S. Benedetto, P., Dubamet, and L. Hanzo, "Symmetric radial basis function network equaliser," In: *NEWCOM-ACoRN Joint Workshop*, 2006.

[6] X. J. Zhu and B. G. Andrew, "Kernel Regression with Order Preferences," *AAAI*, 2007.

[7] T. Yu, *Incorporating prior domain knowledge into inductive machine learning: its implementation in contemporary capital markets*, Ph.D. Thesis, Faculty of Information Technology, University of Technology, Sydney, 2007.

[8] I. Davidson and S. S. Ravi, "Hierarchical clustering with constraints: Theory and practice," *In the 9th European Principles and Practice of KDD*, 2005.

[9] B. G. Hu, Y. Wang, Sh. H. Yang, and H. B. Qu, "How to add transparency to artificial neural networks," (in Chinese with English Abstract), *Pattern Recognition and Artificial Intelligence*, vol. 20, pp. 72-84, 2007.

[10] B. Scholkopf, P. Simard, A. Smola, and V. Vapnik, "Prior knowledge in support vector kernels," *Advances in Neural Information Proceedings Systems*, MIT Press, vol. 10, pp. 640-646, 1998.

[11] C. Goutte and L. K. Hansen, "Regularization with a pruning prior," *Neural Networks*, vol. 10, pp. 1053-1059, 1997.

[12] P. Niyogi, F. Girosi and T. Poggio, "Incorporating prior information in machine learning by creating virtual examples," *Proceedings of the IEEE*, vol. 86, pp. 2196-2209, 1998.

[13] B. G. Hu, H. B. Qu, Y. Wang, and Sh. H. Yang, "A generalized constraint neural network model: Associating partially known relationships for nonlinear regressions," *Information Science*, Vol. 179, pp. 1929-1943, 2009.

[14] W. Chen, Y. Lan, T. Y. Liu, and H. Li, "A unified view on loss functions in learning to rank," *Technical Report, Microsoft Research*, 2009.

[15] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phase for radial-basis-function networks," *Neural Networks*, vol. 14, pp. 439-458, 2001.

[16] M. J. Orr, "Regularization in the selection of radial basis function centres," *Neural Computation*, vol. 7, pp. 606-623, 1995.

[17] M. X. Zhu and D. L. Zhang, "Study on the algorithms of selecting the radial basis function center," *Journal of Anhui University*, vol. 24. pp. 72-78, 2000.

[18] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," In *ICML'05*, pp. 89-96, 2005.

[19] R. Herbrich, K. Obermayer, and T. Graepel, "Large margin rank boundaries for ordinal regression," In *Advances in Large Margin Classifiers*, pp. 115-132, 1999.

[20] X. B. Geng, T. Y. Liu, T. Qin, and H. Li, "Feature selection for ranking," *Proceedings of the 30th Annual International ACM SIGIR Conference*, 2007.

[21] F. Xia, T. Y. Liu, J, Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank - theory and algorithm," In *ICML'08*, 2008.

[22] K. Jarvelin and J. Kekalainen, "IR evaluation methods for retrieving highly relevant documents," *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 41-48, 2000.

[23] K. Jarvelin and J. Kekalainen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, 2002.

[24] V. Vapnik, *Statistical Learning Theory*, Wiley-Interscience, 1998.

[25] M. Bortman and M. Aladjem, "A growing and pruning method for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 20, pp. 1039-1045, 2009.