Proceedings of the NIPS 2005 Workshop on

Learning to Rank

Edited by

Shivani Agarwal
Massachusetts Institute of Technology

Corinna Cortes
Google Research

Ralf Herbrich
Microsoft Research

December 9, 2005
Whistler, BC, Canada

Contents

Acknowledgments

We owe thanks to many people for their contributions to the success of the workshop.

We would like to thank first all the authors who submitted papers and/or abstracts to the workshop. We received many high quality submissions which allowed us to put together an impressive scientific program.

We would also like to express our thanks to our invited speakers: Thorsten Joachims, of Cornell University, for his talk titled "Learning Rankings for Information Retrieval", and Yoram Singer, of The Hebrew University and Google, for his talk titled "Beyond Pair-wise Classification and Regression: Efficient Learning of Preferences by Soft Projections onto Polyhedra".

Finally, we would especially like to thank the members of the program committee for giving up their time to review submissions. Despite the short period of time available, they provided thorough, objective evaluations of their allocated papers, ensuring a high standard of presentations at the workshop. Their names are gratefully listed below.

Shivani Agarwal, Corinna Cortes, and Ralf Herbrich

Program Committee

Calibrated Label-Ranking

Klaus Brinker **Eyke Hüllermeier**
Data and Knowledge Engineering
Otto-von-Guericke-Universität Magdeburg, Germany
{*brinker, huellerm*}*@iti.cs.uni-magdeburg.de*

Abstract

In label-ranking, the goal is to learn a mapping from instances to rankings (total orders) over a fixed set of labels. Hitherto existing approaches to label-ranking implicitly operate on an underlying (utility) scale which is not *calibrated*, that is, which lacks a natural zero point. Since this severely restricts the expressive power of these approaches, we propose a suitable extension of the constraint classification framework to label-ranking. Beyond the general case, we focus on a particular category of ranking problems which admit a representation as a special instance of calibrated label-ranking: In *top label-ranking*, an extension of multilabel classification an ordering is required for the subset of labels relevant for the given instance.

1 Introduction

The increasing trend to consider complex and structured output spaces describes an essential building block in extending the application range of machine learning techniques [1, 2]. In this context, the constraint classification framework [3] provides an elegant technique for solving a wide variety of learning problems, including multiclass, $l-$multilabel and label-ranking learning. More precisely, constraint classification aims at learning a linear (utility) function for each possible *label* (also referred to as *alternative*). The process of evaluating a hypothesis involves sorting the alternatives according to the associated utility values. A substantial drawback of this approach results from the fact that it operates on a non-calibrated utility scale, that is, a scale which lacks a natural zero-point. In other words, the training process only ensures that the *differences* in utility values induce the correct order over the set of alternatives. However, absolute utility values are meaningless in the sense that hypotheses are invariant to positive rescaling and shifting. Therefore, learning problems such as conventional multilabel learning which require a bipartite partition into complementary sets (relevant and non-relevant) do not admit a representation as special instances of constraint classification. This problem also occurs in a related approach to topic-ranking for multi-labeled documents [4] where hypotheses sort the topics according to their relevance for given documents. As a consequence of the underlying non-calibrated utility scale, this approach cannot determine the relevant/non-relevant cutoff, even though this information is specified in the training data.

Our approach avoids the aforementioned drawbacks by working on a calibrated utility scale which contains a natural zero-point and is able to represent bipartite partitions of alternatives. More precisely, we will present a natural extension of the constraint classification

framework which incorporates learning on calibrated scales. The proposed generalization includes the conventional multilabel and various preference learning (ranking) settings as special cases. We are particularly interested in a practically motivated setting that can be seen as an extension of standard multilabel classification, and that we shall refer to as *top label-ranking*: For each input pattern, a ranking must be specified for the subset of alternatives that are relevant for this input (the top-labels).

The remainder of this paper is organized as follows: The subsequent section introduces the aforementioned generalization of the constraint classification framework. In Section 3, we investigate the special case of top label-ranking and present a perceptron-style algorithm which is amenable to analysis in the online mistake bound model.

2 Calibrated constraint classification

Constraint classification [3] provides a general framework for learning in which many common categories of learning problems can be embedded as special cases. Let us embark on a discussion of the most general learning problem considered in constraint classification, viz, label-ranking, and subsequently apply this general setting to represent a variety of standard learning tasks. We assume a finite set of labels to be given by $\mathcal{D} = \{1, \ldots, d\}$. A set of pairwise preferences on $\mathcal{D}$ forms a binary relation and can be represented as $R \subseteq \mathcal{D} \times \mathcal{D}$, where $(r, s) \in R$ is interpreted as alternative r is preferred over alternative s, also denoted by $r \succ s$. In the following, we assume binary relations on $\mathcal{D}$ to be irreflexive, antisymmetric and consistent in the sense that the transitive closure generates a strict partial order, and denote the set of binary relations for which these assumptions hold by $\overline{\mathcal{S}}^{(d)}$. Moreover, a (strict) total order over the finite set $\mathcal{D}$ can be represented by a permutation as there exists a unique permutation τ such that $r \succ s$ iff $\tau(r) > \tau(s)$ ($\tau(r)$ denotes the position of the alternative r in the ranking). Hence, we will identify the set of total preference orders with the set of all permutations $\mathcal{S}^{(d)}$ over $\mathcal{D}$.

We restrict the following discussion to the linear case, i.e., $\mathbb{R}^n$ endowed with the canonical dot product $\langle \cdot, \cdot \rangle$. However, in a natural manner, it is possible to incorporate the concept of kernels. Constraint classification studies the problem of inducing a label-ranking function $h : \mathbb{R}^n \to \mathcal{S}^{(d)}$, $\mathbf{x} \mapsto \operatorname{argsort}_{i=1,\ldots,d} \langle \mathbf{w}_i, \mathbf{x} \rangle$ on the basis of a finite sample of training examples $\{(\mathbf{x}_1, R_1), \ldots, (\mathbf{x}_m, R_m)\} \subset \mathbb{R}^n \times \overline{\mathcal{S}}^{(d)}$. Note that it suffices for training examples to be associated with restricted knowledge on pairwise preferences, whereas the hypotheses are mappings to the set of total orders.

A variety of learning problems can be expressed as special cases of label-ranking by generating appropriate sets of corresponding pairwise preferences in combination with straightforward projections of $\mathcal{S}^{(d)}$ to the specific output space $\mathcal{Y}$ as a post-processing step, among those are the following categories of learning problems:
Multiclass Classification: For each classification example $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ with $\mathcal{Y} = \{1, \ldots, d\}$ a corresponding label-ranking example is defined by $(\mathbf{x}, R(y))$ where $R(y) = \{(y, s) \mid 1 \leq s \leq d,\ s \neq y\}$. The output space $\mathcal{S}^{(d)}$ is projected to the first component.
l-Multilabel Classification: Each input pattern $\mathbf{x}$ is associated with a subset $Y \subset \{1, \ldots, d\}$ of labels ($\mathcal{Y} = \mathfrak{P}(\{1, \ldots, d\})$). The corresponding set of preferences is given by $R(Y) = \{(r, s) \mid r \in Y,\ s \notin Y\}$ and the output space is projected to the first l components.

However, certain classes of learning problems do not admit such an embedding. In particular, conventional multilabel and top label-ranking learning are examples of special relevance in real-world applications. The underlying reason for these drawbacks is that constraint classification (like alternative approaches to label-ranking [2, 5]) implicitly operates on a non-calibrated scale, such that absolute utility values $\langle \mathbf{w}_i, \mathbf{x} \rangle$ do not provide a reasonable basis for defining a bipartite partition of alternatives based on a certain threshold.

In the following, we will propose an extension of constraint classification to so-called *calibrated label-ranking* which operates on a calibrated utility scale with a natural zero-point. The zero-point provides a means to distinguish between the top and the complementary set of alternatives. Let us proceed to a formal definition of the hypothesis space underlying the calibrated constraint classification approach:

Definition 2.1 (Calibrated Linear Sorting Function)**.** *Let* $\mathbf{w}_0 \stackrel{def}{=} \mathbf{0} \in \mathbb{R}^n$ *and suppose we are given weight vectors* $\mathbf{w}_1, \ldots, \mathbf{w}_d \in \mathbb{R}^n$ *of linear functions. Then, the corresponding* calibrated linear sorting function *is defined as*

$$h : \mathbb{R}^n \to \mathcal{S}_0^{(d)}, \quad \mathbf{x} \mapsto \underset{i=0,\ldots,d}{\operatorname{argsort}} \langle \mathbf{w}_i, \mathbf{x} \rangle. \tag{1}$$

Note that $\langle \mathbf{w}_0, \mathbf{x} \rangle = 0$ *for all* $\mathbf{x} \in \mathbb{R}^n$ *and hence the position of the supplemented alternative* 0 *determines the zero-point.*

As an appealing property inherited from the conventional constraint classification framework, the weight vectors defining a calibrated linear sorting function can be learned using standard linear learning algorithms. To this end, we will outline a generalization of the transformation technique proposed in [3]. The required training examples are supplemented with indicator sets defining the (incomplete) bipartite partition of alternatives. Thus, in calibrated label-ranking, we assume a training set to be given by $L = \{(\mathbf{x}_1, R_1, P_1, N_1), \ldots, (\mathbf{x}_m, R_m, P_m, N_m)\} \subset \mathcal{X} \times \overline{\mathcal{S}}^{(d)} \times \mathcal{D} \times \mathcal{D}$, where P_i and N_i denote disjoint sets of alternatives ($P_i \cap N_i = \emptyset$, $P_i \cup N_i \subseteq \mathcal{D}$). We assume consistency of the preferences implicitly given by $(r, s) \in P_i \times N_i$ with the transitive closure of R_i.

Suppose we are given a calibrated-ranking example $(\mathbf{x}, R, P, N)$. Then, two binary training examples are generated for each $(r, s) \in R \cup (P \times N)$, where the original n-dimensional training example is mapped into a nd-dimensional space. The positive example copies the original training vector into the components $(r-1)d+1 \ldots rd$ and its negation into the components $(s-1)d+1 \ldots sd$ of a vector in the new space. The remaining entries are filled with zeros, and the negative example has the same elements with reversed signs. Additionally, for each $r \in P$ a positive example is generated without the negation component and likewise a negative example is constructed for each $r \in N$. The union of all binary examples associated with the original training set is submitted to an arbitrary linear learning algorithm which determines a consistent binary classifier (in the linearly separable case) $h_{\bar{\mathbf{w}}}(\cdot) = \operatorname{sign}(\langle \bar{\mathbf{w}}, \cdot \rangle)$ (with $\bar{\mathbf{w}} \in \mathbb{R}^{nd}$). Furthermore, we consider $\bar{\mathbf{w}}$ as the concatenation of d n–dimensional vectors $\mathbf{w}_1, \ldots, \mathbf{w}_d$ (with $\mathbf{w}_0 \stackrel{\text{def}}{=} \mathbf{0}$) and by this means define a calibrated linear sorting function:

$$h : \mathbb{R}^n \to \mathcal{S}_0^{(d)}, \mathbf{x} \mapsto \underset{i=0,\ldots,d}{\operatorname{argsort}} \langle \mathbf{w}_i, \mathbf{x} \rangle. \tag{2}$$

The expanded sets generated with respect to $R \cup (P \times N)$ encode the constraints ensuring the consistency with respect to the sets of explicit and implicit pairwise preferences. As a particular novel feature of our approach, the additional binary examples related to P and N yield a calibrated separation such that for all examples $(\mathbf{x}, R, P, N) \in L$, alternatives in P correspond to positive and alternatives in N to negative utility values.

Several import categories of learning problems which cannot be solved in the conventional constraint classification framework admit a calibrated representation. Among those are conventional multilabel learning, top label-ranking and a generalization of topic-ranking using multi-labeled data [4] which differentiates between relevant and non-relevant alternatives (see Section 3). In the case of multilabel learning, the conversion process has to be supplemented with a specification of the associated sets of relevant and non-relevant alternatives. For a multilabel example $(\mathbf{x}, Y) \in \mathcal{X} \times \mathfrak{P}(\{1, \ldots, d\})$, these sets are given by $P = Y$ and $N = \{1, \ldots, d\} \backslash Y$, respectively.[1] The calibrated scale provides the basis of

[1] For a given set A, the corresponding power set is denoted by $\mathfrak{P}(A)$.

the projection to the (unordered) set of alternatives having lower rank than the zero-point alternative, i.e., corresponding to positive utility values. In the case of topic-ranking with multi-labeled data, the only modification necessary is to employ the identity projection.

3 Top label-ranking

In this section, we study the problem of learning top label-rankings as a particular example for learning problems in the calibrated constraint classification framework. This category shares a number of similarities with the topic-ranking model for multi-labeled documents proposed in [4] and, moreover, describes a generalization of conventional label-ranking learning [2, 5]. As already stated in the introductory section, top label-ranking distinguishes between the set of relevant and the complementary set of non-relevant alternatives and aims at learning a ranking over the set of relevant alternatives. Non-relevant alternatives are considered to form an equivalence class with respect to the preference order.

We study the most elementary case where training examples are associated with *complete* rankings over the set of relevant alternatives, i.e., associated target objects submitted for training are elements of the considered output space of hypotheses. As mentioned before, the calibrated constraint classification framework is able to employ restricted and incomplete knowledge of preferences. Thus, both the underlying space for target objects in the training data and the output space are identical and will be denoted by $\mathcal{S}^{*(d)}$. Formally, $\mathcal{S}^{*(d)}$ and the elements of the corresponding hypothesis space are defined as follows:

Definition 3.1 (Space of Top Label-Rankings)**.** *Denote by $d \in \mathbb{N}$ the number of alternatives, then the space of all top label-rankings of degree d is defined as*

$$\mathcal{S}^{*(d)} \stackrel{def}{=} \{([y]_1, \ldots, [y]_l) \in \{1, \ldots, d\}^l \mid 1 \leq l \leq d,\, [y]_i \neq [y]_j \quad for \quad i \neq j\}. \tag{3}$$

Note that this encoding assumes the set of relevant alternatives to be implicitly given by $P = \{[y]_1, \ldots, [y]_l\}$ and the non-relevant ones by $N = \mathcal{D} \backslash \{[y]_1, \ldots, [y]_l\}$.

Definition 3.2 (Top Label-Ranking Function)**.** *Suppose we are given weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_d \in \mathbb{R}^n$ of linear functions. Then, the corresponding* top label-ranking function *is defined as*

$$h : \mathbb{R}^n \to \mathcal{S}^{*(d)}, \quad \mathbf{x} \mapsto \underset{i=1,\ldots,d}{\operatorname{argsort}^+} \langle \mathbf{w}_i, \mathbf{x} \rangle \tag{4}$$

where $\operatorname{argsort}^+$ *returns a permutation of $\{i \in \{1, \ldots, d\} \mid \langle \mathbf{w}_i, \mathbf{x} \rangle > 0\}$ where r precedes s if $\langle \mathbf{w}_r, \mathbf{x} \rangle > \langle \mathbf{w}_s, \mathbf{x} \rangle$ (in the case of equality, r precedes s if $r < s$). In the case of exclusively non-positive utility values,* $\operatorname{argsort}^+$ *returns the empty set.*

Top label-ranking can be viewed as a special case of calibrated constraint classification using a suitable projection function to the positive utility alternatives (cf. multilabel classification). In the following, we will discuss this integrated approach which allows for an efficient and elegant solution in detail.

As a first step, we have to study the process of converting rankings to sets of pairwise preferences. Suppose we are given a complete ranking $([y]_1, \ldots, [y]_d)$ on a set of alternatives P. Har-Peled et al. [3] suggest to exploit the special structure of the hypothesis space of linear sorting functions and to convert complete rankings using only adjacent preferences, hence, the corresponding set of preferences is given by $R(y) = \{([y]_i, [y]_{i+1}) \mid i = 1, \ldots, d-1\}$. Note that the transitive closure of $R(y)$ generates the omitted set of preferences. As an alternative option, we can generate all pairwise preferences $R'(y) = \{([y]_i, [y]_j) \mid 1 \leq i < j \leq d\}$. While the first option seems to be favorable as it generates only $d - 1$ preferences whereas the second one leads to $d(d-1)/2$ pairwise preferences, the choice among these options has important consequences: Assume that the binary training set $\bar{L}$ corresponding to a conventional label-ranking problem is not linearly separable and the linear

Algorithm 1 Online calibrated constraint classification

Input:

$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \subset \mathcal{X} \times \mathcal{Y}$ (sequence of training examples)

$R : \mathcal{Y} \to \overline{\mathcal{S}}^{(d)},\ P : \mathcal{Y} \to \mathcal{D},\ N : \mathcal{Y} \to \mathcal{D}$ (embedding functions)

$\pi : \mathcal{S}_0^{(d)} \to \mathcal{Y}$ (projection function)

$\text{loss} : \mathcal{Y} \times \mathcal{Y} \to [0, \Lambda]$ (loss function)

initialize $\mathbf{w}_1^0, \ldots, \mathbf{w}_d^0 \leftarrow \mathbf{0}$, $\mathbf{w}_0^0, \ldots, \mathbf{w}_0^{m+1} \leftarrow \mathbf{0}$
for t=1...,m **do**
  if $\text{loss}(h_t^\pi(\mathbf{x}_t), y_t) = \text{loss}(\pi(\text{argsort}_{i=0,\ldots,d} \langle \mathbf{w}_i^t, \mathbf{x}_t \rangle), y_t) > 0$ **then**
    $\beta_1^t, \ldots, \beta_d^t \leftarrow 0$, $c_t \leftarrow 0$
    for all $(r, s) \in R(y_t) \cup (P(y_t) \times N(y_t))$ **do**
      if $\langle \mathbf{w}_r^t, \mathbf{x}_t \rangle \leq \langle \mathbf{w}_s^t, \mathbf{x}_t \rangle$ **then** $\beta_r^t \leftarrow \beta_r^t + 1$, $\beta_s^t \leftarrow \beta_s^t - 1$, $c^t \leftarrow c^t + 1$
    for all $r \in P(y_t)$ **do**
      if $\langle \mathbf{w}_r^t, \mathbf{x}_t \rangle \leq 0$ **then** $\beta_r^t \leftarrow \beta_r^t + 1$, $c^t \leftarrow c^t + 1$
    for all $r \in N(y_t)$ **do**
      if $\langle \mathbf{w}_r^t, \mathbf{x}_t \rangle \geq 0$ **then** $\beta_r^t \leftarrow \beta_r^t - 1$, $c^t \leftarrow c^t + 1$
    for $i = 1, \ldots, d$ **do**
      $\mathbf{w}_i^{t+1} \leftarrow \mathbf{w}_i^t + \frac{\text{loss}(h_t^\pi(\mathbf{x}_t), y_t)\, \beta_i^t}{c^t} \mathbf{x}_t$
  end if
end for

Output:

$h^\pi(\cdot) = \pi(\text{argsort}_{i=0,\ldots,d} \langle \mathbf{w}_i^{m+1}, \cdot \rangle)$ (calibrated constraint classifier)

learning algorithm, e.g., soft-margin support vector machines, (approximately) minimizes the misclassification error. Then, the conversion process implicitly defines the loss function since the learning algorithm aims at minimizing the number of violated constraints (pairwise preferences). For instance, if the correct ranking is $(1, 2, 3, 4, 5, 6)$ and the hypothesis predicts $(6, 2, 3, 4, 5, 1)$, then the number of corresponding binary misclassifications evaluates to 2 (normalized: $2/5 = 0.4$) in the first case, and 9 (normalized: $9/15 = 0.6$) in the second case. In general, the ratio of the normalized misclassification errors between the second and the first option for the first-last-swap example evaluates to $2 - \frac{3}{d}$ and thus grows with the number of alternatives d. Moreover, minimizing the number of misclassifications when using the pairwise conversion process, is equivalent to minimizing the well-known Kendall τ loss function, which calculates the number of pairwise rank inversions (see [6, 7] for similar expansion approaches). Therefore, we have to be aware of the implicit notion of loss introduced by the particular conversion.

In the case of top label-ranking, the internal expansion offers an analogous alternative as we may reduce the set of constraints associated with P to the single constraint induced by lowest ranked relevant alternative (for N, we have to consider all constraints as no ranking is given among non-relevant alternatives). In a similar manner, this procedure contributes to an implicit definition of loss where the penalty for a relevant alternative being erroneous predicted non-relevant is decreased (except for the lowest ranked relevant alternative).

In the following, we will present an efficient online learning algorithm for calibrated constraint classification which avoids the explicit expansion of examples. It will be sufficient to consider the set of constraints encoded as positive examples (which are associated with sets R) while the origin-symmetric negative examples are not required as the induced updates on weight vectors are identical [3]. Beyond these issues related to computational efficiency, this online algorithm can incorporate a wide variety of loss functions in an *explicit* manner, e.g., in the case of top label-ranking, the generalized Kendall τ and Spearman footrule distance [8]. The cumulative loss associated with this algorithm can be bounded from above using standard perceptron-related techniques (omitted due to space restrictions).

Har-Peled et al. [3] propose an online meta-algorithm for conventional constraint classification which can be generalized to calibrated constraint classification in a natural manner. The particular promotion and demotion values for the weight vectors $\mathbf{w}_i$ are determined using an adaptation of the online algorithm for category ranking presented in [4]. For the sake of simplicity, we assume the existence of embedding *functions* R, P, N which generate the required sets of (pairs of) alternatives $R(y), P(y)$ and $N(y)$ associated with the target objects submitted for training. Technically, the generalization to the more flexible case where examples are individually specified as tuples $(\mathbf{x}, R, P, N)$ is straightforward, but from a conceptual point of view, it requires the definition of a suitable loss function on $\mathcal{Y} \times (\overline{\mathcal{S}}^{(d)} \times \mathcal{D} \times \mathcal{D})$ instead of $\mathcal{Y} \times \mathcal{Y}$ (where $\mathcal{Y}$ denotes the actual output space of the specific problem at hand). The pseudo-code of this algorithm is given in Algorithm 1.

4 Conclusion

We studied the problem of learning label-ranking functions on a calibrated scale with a natural zero point. This setting describes an extension of conventional label-ranking and admits the embedding of a broader variety of practically relevant learning problems as special cases. To this end, we presented an extension of the constraint classification framework and proposed an efficient online learning algorithm amenable to analysis in the mistake bound model. In particular, we studied the problem of learning top label-rankings. Preliminary empirical results, omitted here due to space restriction, are rather promising. These results, just as theoretical results related to the mistake bound model, can be found in an extended version of this paper.[2] As a direction of further research, we plan to conduct an empirical evaluation of our approach investigating the relationship of the implicit notion of error introduced by the constraint encoding process and the explicit loss function incorporated in the online algorithm in detail. Moreover, the underlying concept of introducing a zero-point element can be adapted to the pairwise ranking approach, prompting for both a theoretical and empirical comparison.

Acknowledgements

This research was supported by the German Research Foundation (DFG) and Siemens Corporate Research (Princeton, USA).

References

[1] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[2] Johannes Fürnkranz and Eyke Hüllermeier. Pairwise preference learning and ranking. In *Proceedings of the ECML 2003*, pages 145–156, Cavtat, Croatia, 2003. Springer-Verlag.

[3] Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification: A new approach to multiclass classification and ranking. In *Advances in NIPS 15*, 2002.

[4] Kobi Crammer and Yoram Singer. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3:1025–1058, 2003.

[5] Ofer Dekel, Christopher D. Manning, and Yoram Singer. Log-linear models for label ranking. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in NIPS 16*, pages 497–504, Cambridge, MA, 2004. MIT Press.

[6] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2002.

[7] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. *Advances in Large Margin Classifiers*, chapter Large margin rank boundaries for ordinal regression, pages 115–132. 2000.

[8] Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, and Erik Vee. Comparing and aggregating rankings with ties. In *Proc. 23rd ACM Symposium on Principles of Database Systems (PODS)*, pages 47–58, 2004.

[2] http://wwwiti.cs.uni-magdeburg.de/iti_dke/calibrated.pdf

Ranking as Learning Structured Outputs

Christopher J.C. Burges
Microsoft Research
One Microsoft Way
Redmond, WA 98052-6399
cburges@microsoft.com

Abstract

We propose a new method for learning structured outputs using gradient descent.

1 Introduction

We examine ranking from the point of view of learning structured outputs. To make the discussion concrete we will use the example of ranking search results. There, the task is the following: a query $Q \in \{Q_i : i = 1, \ldots, N_Q\}$ is issued by a user. Q may be thought of as a text string, but it may also contain other kinds of data. The search engine examines a large set of documents, and for each document D, constructs a feature vector $\mathbf{x}(Q, D) \in \mathcal{R}^n$. The feature vector $\mathbf{x}$ is then input to a ranking algorithm $\mathcal{A}$, which outputs a scalar score s: $\mathcal{A} : \mathbf{x} \in \mathcal{R}^n \mapsto s \in \mathcal{R}$. For training, a set of labeled data $\{Q_i, D_{ij}, l_{ij},\ i = 1, \ldots, N_Q,\ j = 1, \ldots, n_i\}$, where n_i is the number of documents returned for the i'th query, is used to minimize a real-valued cost function C. Here the labels l encode the relevance of document D_{ij} with respect to the query Q_i, and take integer values, where for a given query Q, $l_1 > l_2$ means that the document with label l_1 has been judged more relevant to Q than that with label l_2. Once training is complete, the scores s output by $\mathcal{A}$ are used to map feature vectors $\mathbf{x}$ to the reals, where $\mathcal{A}(\mathbf{x}(Q, D_1)) > \mathcal{A}(\mathbf{x}(Q, D_2))$ is taken to mean that, for query Q, document D_1 is to be ranked higher than document D_2.

2 Ranking Measures

In the information retrieval literature, there are many methods used to measure the quality of ranking results. Here we briefly describe four of the simplest. The pair-wise error counts the number of pairs that are in the incorrect order, as a fraction of the maximum possible number of such pairs. The Mean Reciprocal Rank (MRR) applies to the binary relevance task: for a given query, any returned document is labeled 'relevant' or 'not relevant', and if r_i is the rank of the highest ranking relevant document for the i'th query, then the reciprocal rank measure for that query is $1/r_i$, and the MRR is just the reciprocal rank, averaged over queries. 'Winner Takes All' also applies to the binary relevance task: if the top ranked document for a given query is relevant, the WTA cost is zero, otherwise it is one; WTA is again averaged over queries. MRR and WTA have been used, for example, in past TREC evaluations of Question Answering systems. Finally, the normalized discounted cumulative gain measure (NDCG) [9] is a cumulative measure of ranking quality

(so a suitable cost would be $1-$ NDCG). For a given query Q_i the NDCG is computed as $\mathcal{N}_i \equiv N_i \sum_{j=1}^{L} (2^{r(j)} - 1)/\log(1+j)$, where $r(j)$ is the relevance level of the j'th ranked document, and where the normalization constant N_i is chosen so that a perfect ordering would result in $\mathcal{N}_i = 1$. Here L is the ranking level at which the NDCG is computed. The $\mathcal{N}_i$ are then again averaged over the query set.

3 Ranking as Learning Structured Outputs

Many different approaches to learning ranking algorithms have been proposed: from SVMs [6, 10, 3] to Perceptrons [4, 5] to Neural Networks [2, 1]. Are these algorithms solving the right problem? They are certainly attempting to learn an ordering of the data. However, in this Section we argue that, in general, the answer is no. Let's revisit the cost metrics described above. We assume throughout that the documents have been ordered by decreasing score.

These metrics present two key challenges. First, they all depend on not just the output s for a single feature vector $\mathbf{x}$, but on the outputs of all feature vectors, for a given query; for example for WTA, we must compare all the scores to find the maximum. Second, none are differentiable functions of their arguments; in fact they are flat over large regions of parameter space, which makes the learning problem much more challenging. By contrast, note that the algorithms mentioned above have the property that, in order to make the learning problem tractable, they use smooth costs. This smoothness requirement is, in principle, not necessarily a burden, since in the ideal case, when the algorithm can achieve zero cost on the some dataset, using, say, the RankNet cost [1], it has also achieved zero cost using any of the above measures. Hence, the problems that arise from using a simple, smooth approximation to one of the above cost functions, arise because in practice, learning algorithms cannot achieve perfect generalization.

For a concrete example of where using an approximate cost can lead to problems, suppose that we use a smooth approximation to pair-wise error [1], but that what we really want to minimize is the WTA cost. Consider a training query with 1,000 returned documents, and suppose that there are two relevant documents D_1 and D_2, and 998 irrelevant documents, and that the ranker puts D_1 in position 1 and D_2 in position 1000. Then the ranker can reduce the pair-wise error, for that query, by 996 errors, by moving D_2 up to rank 3 and by moving D_1 down to rank 2. However the WTA cost has gone from zero to one. A huge decrease in the pairwise error rate has resulted in the maximum possible increase in the WTA cost.

The need for the ability to handle multivariate costs is not limited to ranking. For example, one measure of quality for document retrieval is the area under the ROC curve, and maximizing this amounts to learning using a multivariate cost, as does using measures that depend on precision and recall [11, 7].

In order to learn using a multivariate, non-differentiable cost function, we propose the following framework, which for the ranking problem we call LambdaRank. We describe the approach in the context of learning to rank using gradient descent. Here, a general multivariate cost function for a given query takes the form $C(s_{ij}, l_{ij})$, where i indexes the query and j indexes a returned document for that query. Thus, in general the cost function may take a different number of arguments, depending on the query (some queries may get more documents returned than others). In general, finding a smooth cost function that has the desired behaviour is very difficult. Take the above WTA example. It is much more important to keep D_1 in the top position than to move D_2 up 997 positions and D_1 down one: the optimal WTA cost is achieved when either D_1 or D_2 is in the top position. Notice how the finite capacity of the learning algorithm is playing a crucial role here. In this particular case, to better approximate WTA, one approach would be to steeply discount

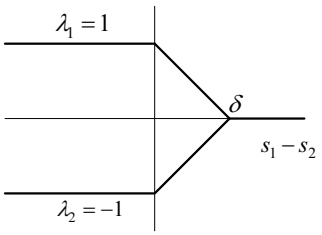

Figure 1: Choosing the λ's for a query with two documents.

errors that occur low in the ranking. Now imagine that C is a smooth approximation to a cost function that accomplishes this, and assume that at the current learning iteration, $\mathcal{A}$ produces an ordering for a given Q where D_1 is in position 2 and D_2 is in position 1000. Then if $s_i \equiv \mathcal{A}(\mathbf{x}_i),\ \ i = 1, 2$, we require that

$$|\frac{\partial C}{\partial s_1}| \gg |\frac{\partial C}{\partial s_2}| \tag{1}$$

Notice that we've captured a desired property of C by imposing a constraint on its derivatives. The idea of LambdaRank is to extend this by replacing the requirement of specifying C itself, by the task of specifying its derivative with respect to each s_j, $j = 1, \ldots, n_i$, for each query Q_i. Those derivatives can then be used to train $\mathcal{A}$ using gradient descent, just as the derivatives of C normally would be. The point is that it can be much easier, given an instance of a query and its ranked documents, to specify how you would like those documents to move, in order to reduce a non-differentiable cost, than to specify a smooth approximation of that (multivariate) cost. As a simple example, consider a single query with just two returned documents D_1 and D_2, and suppose they have labels $l_1 = 1$ (relevant) and $l_2 = 0$ (not relevant), respectively. We imagine that there is some $C(s_1, l_1, s_2, l_2)$ such that

$$\frac{\partial C}{\partial s_1} = -\lambda_1(s_1, l_1, s_2, l_2) \tag{2}$$

$$\frac{\partial C}{\partial s_2} = -\lambda_2(s_1, l_1, s_2, l_2) \tag{3}$$

We would like the λ's to take the form shown in Figure 1, for some chosen margin $\delta \in \mathcal{R}$: thinking of the documents as lying on a vertical line, where higher scores s correspond to higher points on the line, then D_1 (D_2) gets a constant gradient up (or down) as long as it is in the incorrect position, and the gradient goes smoothly to zero until the margin is achieved. Thus the learning algorithm $\mathcal{A}$ will not waste capacity moving D_1 further away from D_2 if they are in the correct position by more than δ, and having nonzero δ ensures robustness to small changes in the scores s_i. Letting $x \equiv s_1 - s_2$, the λ's may be written

$$x < 0 \ : \ \lambda_1 = 1 = -\lambda_2 \tag{4}$$

$$0 \leq x \leq \delta \ : \ \lambda_1 = \delta - x = -\lambda_2 \tag{5}$$

$$x > \delta \ : \ \lambda_1 = \lambda_2 = 0 \tag{6}$$

In this case a corresponding cost function exists:

$$x < 0 \quad : \quad C(s_1, l_1, s_2, l_2) = s_2 - s_1 \tag{7}$$

$$0 \leq x \leq \delta \quad : \quad C(s_1, l_1, s_2, l_2) = \frac{1}{2}(s_1 - s_2)^2 - \delta(s_1 - s_2) \tag{8}$$

$$x > \delta \quad : \quad C(s_1, l_1, s_2, l_2) = -\frac{1}{2}\delta^2 \tag{9}$$

Note that in addition the Hessian of C is positive semidefinite, so the cost function takes a unique minimum value (although the s's for which C attains its minimum are not unique). In general, when the number of documents for a given query is much larger than two, and where the rules for writing down the λ's will depend on the scores, labels and ranks of all the documents, then C can become prohibitively complicated to write down explicitly.

There is still a great deal of freedom in this model, namely, how to choose the λ's to best model a given (multivariate, non-differentiable) cost function. Let's call this choice the λ-function. We will not explore here how, given a cost function, to find a particular λ-function, but instead will answer two questions which will help guide the choice: first, for a given choice of the λ's, under what conditions does there exists a cost function C for which they are the negative derivatives? Second, given that such a C exists, under what conditions is C convex? The latter is desirable to avoid the problem that local minima in the cost function itself will present to any algorithm used for training $\mathcal{A}$. To address the first question, we can use a well-known result from multilinear algebra [12]:

Theorem (Poincaré Lemma): If $S \subset \mathcal{R}^n$ is an open set that is star-shaped with respect to 0, then every closed form on S is exact.

Note that since every exact form is closed, it follows that on an open set that is star-shaped with respect to 0, a form is closed if and only if it is exact. Now for a given query Q_i and corresponding set of returned D_{ij}, the n_i λ's are functions of the scores s_{ij}, parameterized by the (fixed) labels l_{ij}. Let dx^i be a basis of 1-forms on $\mathcal{R}^n$ and define the 1-form

$$\boldsymbol{\lambda} \equiv \sum_i \lambda_i dx^i \tag{10}$$

Then assuming that the scores are defined over $\mathcal{R}^n$, the conditions for the theorem are satisfied and $\boldsymbol{\lambda} = dC$ for some function C if and only if $d\boldsymbol{\lambda} = 0$ everywhere. Using classical notation, this amounts to requiring that

$$\frac{\partial \lambda_i}{\partial s_j} = \frac{\partial \lambda_j}{\partial s_i} \quad \forall i, j \tag{11}$$

Thus we have a simple test on the λ's to determine if there exists a cost function for which they are the derivatives: the Jacobian (that is, the matrix $J_{ij} \equiv \partial \lambda_i / \partial s_j$) must be symmetric. Furthermore, given that such a cost function C does exist, the condition that it be convex is that the Jacobian be positive semidefinite everywhere. Under these constraints, the Jacobian is beginning to look very much like a kernel matrix! However, there is a difference: the value of the i'th, j'th element of a kernel matrix depends on two vectors $\mathbf{x}_i$, $\mathbf{x}_j$ (where for example $\mathbf{x} \in \mathcal{R}^d$ for some d, although in general they may be elements of an abstract vector space), whereas the value of the i'th, j'th element of the Jacobian depends on all of the scores s_i.

For choices of the λ's that are piecewise constant, the above two conditions (symmetric and positive semidefinite[1]) are trivially satisfied. For other choices of symmetric J,

[1]Some authors define the property of positive semi-definiteness to include the property of symmetry: see [8].

positive definiteness can be imposed by adding regularization terms of the form $\lambda_i \mapsto \lambda_i + \alpha s_i, \;\; \alpha_i > 0$, which amounts to adding a positive constant along the diagonal of the Hessian.

Finally, we observe that LambdaRank has a clear physical analogy. Think of the documents returned for a given query as point masses. Each λ then corresponds to a force on the corresponding point. If the conditions of Eq. (11) are met, then the forces in the model are conservative, that is, the forces may be viewed as arising from a potential energy function, which in our case is the cost function. For example, if the λ's are linear in the outputs s, then this corresponds to a spring model, with springs that are either compressed or extended. The requirement that the Jacobian is positive semidefinite amounts to the requirement that the system of springs have a unique global minimum of the potential energy, which can be found from any initial conditions by gradient descent (this is not true in general, for arbitrary systems of springs).

References

[1] C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, Learning to Rank Using Gradient Descent, in *ICML 22*, 2005

[2] R. Caruana, S. Baluja, and T. Mitchell, Using the Future to "sort out" the Present: Rankprop and Multitask Learning for Medical Risk Evaluation, *NIPS 8*, 1996.

[3] W. Chu and S.S. Keerthi, New Approaches to Support Vector Ordinal Regression, in *ICML 22*, 2005

[4] K. Crammer and Y. Singer, Pranking with Ranking, in *NIPS 14*, 2002

[5] E.F. Harrington, Online Ranking/Collaborative Filtering Using the Perceptron Algorithm, in *ICML 20*, 2003

[6] R. Herbrich, T. Graepel, and K. Obermayer, Large Margin Rank Boundaries for Ordinal Regression, in A.J. Smola et al., Editors, *Advances in Large Margin Classifiers*, 2000

[7] A. Herschtal and B. Raskutti, Optimizing Area Under the ROC Curve Using Gradient Descent, in *ICML 21*, 2004.

[8] R.A. Horn and C.R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[9] K. Jarvelin and J. Kekalainen, IR Evaluation Methods for Retrieving Highly Relevant Documents, in *Proc. 23rd ACM SIGIR*, 2000.

[10] T. Joachims, Optimizing Search Engines Using Clickthrough Data, in David Hand, Daniel Keim, and Raymond Ng, editors, *Proc. 8th SIGKDD*, 2002.

[11] T. Joachims, A Support Vector Method for Multivariate Performance Measures, *ICML 22*, 2005.

[12] M. Spivak, *Calculus on Manifolds*, Addison-Wesley, 1965.

Exploiting Hyperlinks to Learn a Retrieval Model

David Grangier **Samy Bengio**
IDIAP Research Institute, Martigny, Switzerland,
Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
{grangier,bengio}@idiap.ch

Abstract

Information Retrieval (IR) aims at solving a ranking problem: given a query q and a corpus C, the documents of C should be ranked such that the documents relevant to q appear above the others. This task is generally performed by ranking the documents $d \in C$ according to their similarity with respect to q, $sim(q, d)$. The identification of an effective function $a, b \rightarrow sim(a, b)$ could be performed using a large set of queries with their corresponding relevance assessments. However, such data are especially expensive to label, thus, as an alternative, we propose to rely on hyperlink data which convey analogous semantic relationships. We then empirically show that a measure sim inferred from hyperlinked documents can actually outperform the state-of-the-art *Okapi* approach, when applied over a non-hyperlinked retrieval corpus.

1 Introduction

Information Retrieval (IR) consists in finding documents that are relevant to a given query in a large corpus (e.g. more than $100,000$ documents). This task is generally formulated as a ranking problem: given a query q and a set of documents D, an IR system should output a document ranking in which the relevant documents appear above non-relevant ones. In order to achieve such a goal, a common approach consists in ranking the documents $d \in D$ according to their similarity $sim(q, d)$ with respect to the query q [1]. Hence, the identification of a reliable measure of the semantic similarity between text items is of crucial importance in IR. In fact, such a measure sim should ideally compare sequences of terms, referred to as documents and queries in this case, such that

$$\forall q, \forall d^+ \in R(q), \forall d^- \notin R(q), sim(q, d^+) - sim(q, d^-) > 0, \quad (1)$$

$R(q)$ being the set of documents which are relevant to q. This property actually ensures that relevant documents are ranked above non-relevant ones for any query.

The selection of an appropriate measure of similarity could hence be performed through the optimization of a criterion related to (1) over some training data [2, 5]. However, such a process would require a large set of labeled queries for training (i.e. queries with the corresponding relevance set) which are expensive to obtain [1]. As an alternative, we propose to identify an effective measure from already available hyperlinked data D_{train} that can then be applied over any IR corpus D_{test}, with or without hyperlinks.

This approach relies on hyperlinks for training, since such data contain information about the semantic proximity of documents which are close to the document/query relationships provided by relevance assessments. In fact, it has been observed [4] that, in most cases, a

document d is semantically closer to a document l^+, hyperlinked with d, than to a document l^-, not hyperlinked with d:

$$\forall d \in D_{train}, \forall l^+ \in L(d), \forall l^- \notin L(d), sim(d, l^+) - sim(d, l^-) > 0, \quad (2)$$

where $L(d)$ refers to the set of documents linked with d (i.e. the documents referring to d and the documents referred to by d). This kind of relationship is hence analogous to relevance assessments which state that a query q is semantically closer to a document d^+, relevant to q, than to a document d^-, not relevant to q (1).

Our task is hence to identify a measure sim that would ideally verify (2). For that purpose, we introduce a parameterized similarity measure sim_θ and a cost C which penalizes the parameters θ for which a large number of constraints (2) are not verified. The parameter θ^* that minimizes C is then selected through stochastic gradient descent (see Section 2). The function sim_{θ^*} inferred with this approach has then been compared with the state-of-the-art *Okapi* matching [6] over a benchmark IR corpus (TREC-9 queries over the TDT-2 documents). The performance of our approach is shown to outperform *Okapi* with respect to various IR measures (Precision at top 10, P10, Average Precision, AvgP, and Break-Even Point, BEP), the relative improvement being greater than 10% for all measures (see Section 4).

The remainder of this paper is organized as follows, Section 2 describes the proposed model, *LinkLearn*, Section 3 compares this model with alternative approaches, Section 4 presents IR experiments to assess the effectiveness of our approach, finally, Section 5 draws some conclusions.

2 The LinkLearn Model

In this Section, we describe the *LinkLearn* model: first, the parameterization is introduced and then, the training procedure is described.

Model Parameterization
LinkLearn relies on a parametric function sim_θ to compute the similarity between text items. To introduce such a function, we first present how query/document similarity is computed in ad-hoc retrieval systems and we then define a parameterized measure inspired from these approaches.

The Vector Space Model (VSM) is the most common framework to compare text items in IR systems. In this context, each document d is first *indexed* with a vocabulary-sized vector,

$$d = (d_1, \ldots, d_V),$$

where d_i is the weight of term i in document d and V is the vocabulary size. Then, the dot product between such vectors is then used to assess the document similarity. This VSM approach is also often referred to as the *bag-of-words* model, as term ordering is not taken into account. The weights d_i are generally computed as an a-priori defined function of some features of i and d, such as $tf_{i,d}$ the number of occurrences of i in d, df_i the number of documents of D_{train} containing term i, l_d the length of document d (i.e. the total number of term occurrences in d). For instance, the most common weighting function, *Okapi BM25* [6], computes such a weight as

$$d_i = \frac{(K+1) \cdot tf_{i,d} \cdot idf_i}{K \cdot ((1-B) + B \cdot (l_d/L)) + tf_{i,d}},$$

where idf_i is defined as $\log(N/df_i)$, N is the total number of documents in D_{train}, L is the mean of l_d over D_{train}, and K, B are hyperparameters to select.

We hence adopt a similar approach to parameterize our model. In our case, the weight of a term in a document is computed as,

$$d_i^\theta = f_\theta(tf_{i,b}, idf_i, l_b),$$

where f_θ is the product of the outputs of three single-output Multi-Layer Perceptrons (MLP),

$$f_\theta : x, y, z \rightarrow MLP_{\theta_1}(x) \cdot MLP_{\theta_2}(y) \cdot MLP_{\theta_3}(z), \tag{3}$$

and $\theta = [\theta_1, \theta_2, \theta_3]$ corresponds to the MLP parameters. This hence leads to the following parameterized measure of similarity,

$$sim_\theta : a, b \rightarrow \sum_{i=1}^{V} f_\theta(tf_{i,a}, idf_i, l_a) \cdot f_\theta(tf_{i,b}, idf_i, l_b).$$

This measure therefore only relies on simple features of term occurences which makes it *vocabulary-independent*, i.e. the learned parameters are not linked to a specific term set and the function sim inferred from one corpus can therefore be applied to another corpus, possibly indexed with a different vocabulary (e.g. in Section 4, for TREC experiments, training and testing are performed using vocabulary extracted from different corpora).

The proposed parameterization (3) involves 3 different MLPs, each one having a real valued input, which is a limitation with respect to a model where function f would be a unique MLP with a 3-dimensional input. Such a simplification is however necessary in order to apply the model over large corpora since it significantly reduces the required computational cost for both training and testing: instead of evaluating an MLP function for all triplets $\forall d, i, (tf_{d,i}, idf_i, l_d)$, it should only be evaluated for each possible value of $tf_{d,i}$, idf_i and l_d. In Section 4, the number of MLP evaluations would for instance have been $\sim 1,000$ times greater with a single MLP. Moreover, the experimental results show that this simplified parameterization does not prevent our model from reaching good performance.

Model Criterion and Training

This Section describes how the parameter vector θ of the function sim_θ is selected such that most constraints of (2) are respected. For that purpose, we introduce a cost C related to (2) that can be minimized through stochastic gradient descent.

A simple cost to minimize in this context could be the number of constraints which are not verified,

$$C^{0/1} = \sum_{d \in D_{train}} C_d^{0/1}, \tag{4}$$

$$\text{where} \qquad C_d^{0/1} = \sum_{l^+, l^- \in L(d) \times \overline{L(d)}} I\{sim_\theta(d, l^+) - sim_\theta(d, l^-) < 0\} \tag{5}$$

and $I\{\cdot\}$ is the indicator function ($I\{c\} = 1$ if c is true and 0 otherwise).

However, similarly to the $0/1$ loss in the case of classification problems, this cost is obviously not suitable for gradient descent (i.e. its gradient is null everywhere). We hence propose to minimize an upper bound of this quantity:

$$C = \sum_{d \in D_{train}} C_d, \tag{6}$$

$$\text{where} \qquad C_d = \sum_{l^+, l^- \in L(d) \times \overline{L(d)}} |1 - sim_\theta(d, l^+) + sim_\theta(d, l^-)|_+ \tag{7}$$

and $x \rightarrow |x|_+$ is x if $x > 0$, 0 otherwise. This cost is actually an upper bound of $C^{0/1}$ since $\forall x, |1 - x|_+ \geq I\{x < 0\}$. C is then minimized through stochastic gradient descent, i.e. we iteratively pick documents in D_{train} and update θ according to $\partial C_d / \partial \theta$. The hyperparameters of the model (i.e. the number of hidden units in the MLPs, the number of training iterations and the learning rate) are selected through cross-validation (see Section 4).

The use of C has two main advantages: from a theoretical perspective, the minimization of C can be interpreted as margin maximization [3]. Moreover, from a practical point of

view, the gradient $\partial C_d/\partial\theta$ is especially inexpensive to compute since

$$1 - sim_\theta(d, l^+) + sim_\theta(d, l^-) < 0 \Rightarrow \frac{\partial}{\partial\theta}|1 - sim_\theta(d, l^+) + sim_\theta(d, l^-)|_+ = 0.$$

This effectiveness aspect is crucial for training over large datasets, giving to ***LinkLearn*** a scalability advantage over alternative approaches, as explained in the following.

3 Related Works

The inference of document similarity measures (or equivalently document distance metrics) from a set of proximity constraints P_{train} of type

"document a is closer to document b than it is to document c,"

is a recent research topic in Machine Learning. In the following, two alternative models are described: ***Ranking SVM***, a Support Vector Machine approach, and ***RankNet***, a model based on MLP and gradient descent optimization.

Ranking SVM [7] is a distance learning model: it aims at identifying d_w,

$$d_w : x, y \rightarrow \sqrt{\sum_{i=1}^{V} w_i(x_i - y_i)^2},$$

where $\forall i, w_i > 0$, from the constraint set P_{train}:

$$\forall (a, b, c) \in P_{train}, d_w(a, b) < d_w(a, c).$$

As a distance is always positive, the constraints can be reformulated as,

$$\forall (a, b, c) \in P_{train}, d_w(a, c)^2 - d_w(a, b)^2 > 0.$$

To ensure good generalization performance, a margin maximization approach is then adopted, leading to the following problem,

$$\begin{array}{lll} \min\limits_{w,\xi} & \|w\|_2 \quad + \quad C \sum\limits_{(a,b,c)\in P_{train}} \xi_{a,b,c} & \\ s.t. & \left\{ \begin{array}{l} \forall (a, b, c) \in P_{train}, d_w(a, c)^2 - d_w(a, b)^2 \geq 1 - \xi_{a,b,c} \\ \forall (a, b, c) \in P_{train}, \xi_{a,b,c} \geq 0 \\ \forall i = 1 \ldots V, w_i \geq 0. \end{array} \right. \end{array} \quad (8)$$

where C is an hyperparameter that control the trade-off between the margin size and the number of non-verified constraints. Such a model has shown to be effective empirically: e.g. it has notably been used to combine different search engine outputs [5]. However, the resolution of (8) through quadratic optimization becomes computationally costly as the training set size $|P_{train}|$ grows, i.e. $\sim O(|P_{train}|^p), 2 < p \leq 3$, making gradient descent approaches like ***LinkLearn*** or ***RankNet*** a suitable alternative for large constraint sets.

RankNet [2] is a gradient based approach to similarity measure learning. Like ***ranking SVM*** and ***LinkLearn***, this model is also trained from a set proximity constraints P_{train},

$$\forall (a, b, c) \in P_{train}, sim(a, b) > sim(a, c).$$

In this case, each $(a, b, c) \in P_{train}$ is additionally labeled with $p_{a,b,c}$, the probability that constraint (a, b, c) is actually true. This allows for including some confidence information about the training constraints while not preventing to use a set P_{train} without probability (i.e. in this case, it can be assumed that $\forall (a, b, c) \in P_{train}, p_{a,b,c} = 1$).

RankNet relies on some feature vector[1] $\phi(a, b)$ to compute the similarity between text items a and b,

$$sim_\theta(a, b) = MLP_\theta(\phi(a, b))$$

The parameter vector θ is then identified from P_{train} through the minimization of the cross-entropy (CE) criterion:

$$C^{(CE)} = \sum_{(a,b,c)\in P_{train}} C^{(CE)}_{a,b,c}, \tag{9}$$

$$\text{where} \quad C^{(CE)}_{a,b,c} = -p_{a,b,c} \log o_{a,b,c} - (1 - p_{a,b,c}) \log(1 - o_{a,b,c}) \tag{10}$$

$$\text{and} \quad o_{a,b,c} = \frac{\exp(sim_\theta(a, b) - sim_\theta(a, c))}{1 + \exp(sim_\theta(a, b) - sim_\theta(a, c))}. \tag{11}$$

Like for *LinkLearn*, this cost can then be minimized through gradient descent optimization. *RankNet* and *LinkLearn* approaches are hence close: the use of gradient descent allows for their application over large training sets. Moreover, they could be applied with any differentiable function sim_θ which enables to easily include some a-priori knowledge about document similarity measures.

These two models are however not identical. On one hand, *RankNet* allows for the assignment of different confidence levels for the proximity constraints (through $p_{a,b,c}$), which can be advantageous in the case where the constraints come from several annotators that may disagree. On the other hand, *LinkLearn* cost allows for efficient gradient computation (see Section 2), which makes it suitable for large training set (e.g. in next Section, *LinkLearn* has been trained over $\sim 10^{11}$ constraints).

4 Experiments and Results

In this Section, we assess the performance of *LinkLearn* according to the following experimental setup: the model is first trained over the *Wikipedia* hyperlinked corpus and the inferred measure sim_{θ^*} is then used to rank the documents of *TDT-2* corpus with respect to *TREC-9* ad-hoc queries. The IR performance over this test set is then compared with respect to the state-of-the-art *Okapi* approach.

Training over Wikipedia Corpus
The Wikipedia corpus[2] consists of encyclopedia articles, each article referring to other related articles using hyperlinks. To train *LinkLearn*, two subsets D_{train} and D_{valid} of $\sim 150,000$ documents have been randomly extracted from the whole dataset ($\sim 450,000$ documents) such that no document belongs to both sets. The hyperlinks which does not start and end in the same subset have been removed, resulting in an average of 13.4 and 12.5 links per documents for D_{train} and D_{valid}. The D_{train} set is used for gradient descent (i.e. C is minimized over this set) and D_{valid} is used to select the model hyperparameters. In order to have an estimate of the IR performance on D_{valid}, the following artificial retrieval task is introduced: each document $d \in D_{valid}$ is considered to be a query whose relevant documents are the documents linked with d and average precision is measured for this task (Figure 1 reports this measurement during training).

Evaluation with TREC-9 queries
In this Section, *LinkLearn* and *Okapi* are compared on TREC-9 queries for the TDT-2 corpus[3]. The TDT-2 corpus contains $24,823$ documents and there are 50 TREC-9 queries, each query having, on average, 13.2 relevant documents. For *LinkLearn*, no re-training

[1] We do not describe ϕ since it has only been briefly presented in the original description of *RankNet* [2].

[2] Wikipedia corpus and documentation are available at download.wikimedia.org.

[3] TREC data and documentation are available at trec.nist.gov.

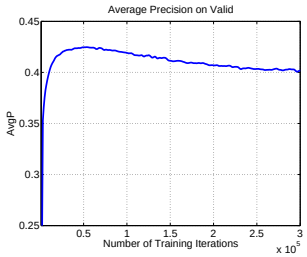

Figure 1: Validation Performance during Training
This plot depicts validation performance up to $300,000$ iterations but early stopping criterion has actually stopped training before over-fitting on the $AvgP$ curve (i.e. after $54,000$ iterations).

Table 1: Retrieval Results over TDT-2/TREC-9 data

	Okapi	*LinkLearn*
P10	38.8%	43.2% $(+11\%)$
BEP	30.3%	35.2% $(+16\%)$
AvgP	29.3%	34.5% $(+18\%)$

or adaptation have been performed. The *LinkLearn* measure inferred from Wikipedia has directly been applied as a query/document matching measure to TDT-2/TREC-9. For *Okapi*, the hyperparameters K, B have been selected through cross-validation over TREC-8 queries. To assess the IR performances of both methods, Precision at top 10, $P10$, Average Precision, $AvgP$, and Break-Even Point, BEP results are reported in Table 1. According to all measures, *LinkLearn* performs better than *Okapi* and the relative improvement is more than 10% in all cases.

5 Conclusions

In this paper, we introduced *LinkLearn*, a gradient descent approach to derive a document similarity measure from a hyperlinked training corpus: the measure is selected such that, in most cases, a document is considered more similar to the documents with which it is linked than to the other documents. The inferred measure can then be applied to compare any text items with or without hyperlinks. In particular, a measure learned with *LinkLearn* over an encyclopedia corpus (Wikipedia) has shown to outperform state-of-the-art *Okapi* matching measure when used to compare documents and queries in an IR ranking problem (TDT-2/TREC-9).

Acknowledgments:
This work has been performed with the support of the Swiss NSF through the NCCR–IM2 project. It was also supported by the PASCAL European Network of Excellence, funded by the Swiss OFES.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[2] C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML*, 2005.

[3] R. Collobert and S. Bengio. Links between perceptrons, MLPs and SVMs. In *ICML*, 2004.

[4] B. D. Davison. Topical locality in the web. In *SIGIR*, 2000.

[5] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, 2002.

[6] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *TREC*, 1994.

[7] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *NIPS*, 2003.

Generalization Bounds for k-Partite Ranking

Shyamsundar Rajaram
Beckman Institute
University of Illinois
Urbana, IL 61801, USA
rajaram1@ifp.uiuc.edu

Shivani Agarwal
Computer Science & AI Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
shivani@csail.mit.edu

Abstract

We study generalization properties of ranking algorithms in the setting of the k-*partite* ranking problem. In the k-partite ranking problem, one is given examples of instances labeled with one of k ordered 'ratings', and the goal is to learn from these examples a real-valued ranking function that ranks instances in accordance with their ratings. This form of ranking problem arises naturally in a variety of applications and, formally, constitutes a generalization of the bipartite ranking problem that has recently been studied. We start by defining notions of ranking error suitable for measuring the quality of a ranking function in the k-partite setting. We then give distribution-free probabilistic bounds on the expected error of a ranking function learned by a k-partite ranking algorithm.

1 Introduction

Consider the following scenario. Kim, an avid reader, is always on the look-out for good books to read. She has read several books in the last few years, and each time she reads a book, she sums up her opinion on the book in the form of a 'rating', which can range from one star (poor) to five stars (excellent). She wonders if, based on these past ratings, someone could generate for her an ordered list of all the books at her local library, such that books she is likely to give a high rating to appear at the top of the list, while books she is likely to give a lower rating to appear at the bottom of the list. She could then use this list as a guide in selecting her next book.

Kim's problem is clearly an instance of a learning problem. How can this problem be formalized? There is an instance space $\mathcal{X}$ (which in Kim's case is the space of all books), and there is a set $\mathcal{Y}$ of k ordered 'ratings', which we can take to be $\mathcal{Y} = \{1, \ldots, k\}$, with k representing the highest rating and 1 the lowest (in Kim's case, $k = 5$). The input to the problem is a finite sequence of labeled training examples $S = ((x_1, y_1), \ldots, (x_M, y_M)) \in (\mathcal{X} \times \mathcal{Y})^M$. What form should the output of a learning algorithm for this problem take?

We could ask that a learning algorithm predict ratings of new instances. The goal of the algorithm would then be to learn from S a function $h : \mathcal{X} \to \mathcal{Y}$; given a new instance $x \in \mathcal{X}$, the algorithm would predict the rating $h(x)$. There could be different ways to measure the quality of such a function h. One possibility is to simply count an error if $h(x)$ differs from the true rating y of x; the error of h with respect to the training sample S would then be

$$\widehat{L}_0(h; S) \quad = \quad \frac{1}{M} \sum_{i=1}^{M} \mathbf{I}_{\{h(x_i) \neq y_i\}} \,, \tag{1}$$

where $\mathbf{I}_{\{\cdot\}}$ denotes the indicator variable whose value is one if its argument is true and zero otherwise. This, however, reduces the problem to one of multi-class classification, in which the k ratings are treated simply as k 'classes'. Such a formulation ignores the order over the ratings: predicting four stars instead of five incurs the same penalty as predicting two stars instead of five. Another possibility is to weigh the error by the difference between the predicted rating $h(x)$ and the true rating y; in this case, the error of h w.r.t. S would be

$$\widehat{L}_1(h;S) \quad = \quad \frac{1}{M}\sum_{i=1}^{M}|h(x_i)-y_i|\,. \tag{2}$$

This corresponds to ordinal regression, and comes closer to fitting our problem.

However, remember that what Kim wants is actually an ordered list of books; in other words, the desired output is an ordering over the instances in $\mathcal{X}$. We could construct an ordering over $\mathcal{X}$ using a function $h : \mathcal{X}\rightarrow\mathcal{Y}$, that predicts ratings of instances in $\mathcal{X}$, by placing instances that are given a higher rating by h above instances that are given a lower rating by h, breaking ties at random; this would place instances predicted to have rating k at the top (in an arbitrary order), followed by instances predicted to have rating $k-1$ (again in an arbitrary order), and so on. A more natural approach, however, is to ask that a learning algorithm output directly an ordering over the instance space. Formally, we require that the algorithm learn from S a real-valued ranking function $f : \mathcal{X}\rightarrow\mathbb{R}$ that assigns scores to instances and thereby induces an ordering or *ranking* over $\mathcal{X}$: an instance $x \in \mathcal{X}$ is ranked higher by f than an instance $x' \in \mathcal{X}$ if $f(x) > f(x')$, and lower if $f(x) < f(x')$.

What would be a good way to measure the quality of a ranking function f? To answer this question, let us consider the bipartite ranking problem which has recently received some attention [7, 1]. In the bipartite ranking problem, instances in the space $\mathcal{X}$ come from two categories, 'positive' and 'negative'. The learner is given a finite number of examples of instances labeled as positive or negative, and the goal is to learn a ranking function $f : \mathcal{X}\rightarrow\mathbb{R}$ that ranks positive instances higher than negative ones. It is easy to see that this problem corresponds exactly to our problem in the special case when $k = 2$, with positive examples viewed as having the higher rating, 2, and negative examples the lower rating, 1. If n_1, n_2 denote the number of examples in the training sample S with ratings 1 and 2, respectively, then the bipartite ranking error (w.r.t. S) of a ranking function f, used to measure the quality of a ranking function in the bipartite setting, can be expressed as

$$\widehat{R}(f;S) \quad = \quad \frac{1}{n_1 n_2}\sum_{\{i:y_i=1\}}\sum_{\{j:y_j=2\}}\lambda(f,x_j,x_i)\,, \tag{3}$$

where

$$\lambda(f,x,x') \quad = \quad \mathbf{I}_{\{f(x)<f(x')\}} + \frac{1}{2}\mathbf{I}_{\{f(x)=f(x')\}}\,. \tag{4}$$

The bipartite ranking error effectively counts an error each time an instance of rating 2 is ranked lower by f than an instance of rating 1 (assuming ties are broken at random). To measure the quality of a ranking function in the general case ($k \geq 2$), we would like to use a quantity that extends naturally the bipartite ranking error above; in particular, we would like the quantity to reduce to the bipartite ranking error in the case $k = 2$.

One way to do this is to count a ranking error each time an instance of a higher rating is ranked lower by f than an instance of a lower rating:

$$\widehat{R}_0(f;S) \quad = \quad \frac{1}{C(S_Y)}\sum_{l=1}^{k-1}\sum_{m=l+1}^{k}\sum_{\{i:y_i=l\}}\sum_{\{j:y_j=m\}}\lambda(f,x_j,x_i)\,, \tag{5}$$

where we have decomposed S into $S_X = (x_1,\ldots,x_M) \in \mathcal{X}^M$ and $S_Y = (y_1,\ldots,y_M) \in \mathcal{Y}^M$, and taken $C(S_Y) = \sum_{l=1}^{k-1}\sum_{m=l+1}^{k} n_l n_m$, with n_l denoting the number of ratings in S_Y equal to l. However, this quantity does not adequately distinguish ranking errors

made between instances of ratings 4 and 5 from ranking errors made between instances of ratings 2 and 5. Such distinctions can be taken into account by measuring the error of f as

$$\widehat{R}_1(f;S) \;=\; \frac{1}{C(S_Y)} \sum_{l=1}^{k-1} \sum_{m=l+1}^{k} \sum_{\{i:y_i=l\}} \sum_{\{j:y_j=m\}} (m-l)\, \lambda(f, x_j, x_i)\,. \qquad (6)$$

If we represent the training sample S as a graph in which there is a vertex for each instance in S and an edge between each pair of vertices for which a mis-ranking of the corresponding pair of instances would contribute a non-zero error term, then under any of the above two forms of ranking error (Eqs. (5) and (6)), the resulting graph is k-partite. For this reason, we refer to this form of ranking problem as the *k-partite* ranking problem.

The problem of ranking, and the related problem of ordinal regression, have recently begun to be studied in machine learning[1] [5, 8, 6, 7, 11, 1, 4]. In the most general setting of the ranking problem [5, 7], the learner is given training examples consisting of ordered pairs of instances, each labeled with a ranking preference that indicates the importance of ranking that pair correctly, and the goal is to learn from these examples a real-valued ranking function that ranks future instances accurately. The bipartite setting of the ranking problem is perhaps one of the simplest, and several theoretical results have recently been derived for it [7, 1, 2, 3]. The k-partite setting described above is a natural extension of the bipartite setting that encompasses a wider range of applications; these include, for example, the book ranking problem described earlier, as well as many other problems that involve ranking based on ordered ratings, such as ranking movies or ranking music albums.

An important question for ranking algorithms, as for all learning algorithms, concerns generalization ability: how well does the empirical error of a learned ranking function, with respect to the training sample from which it is learned, generalize to its expected error on future data? This question has been addressed recently for the bipartite setting in [7, 1, 2], and, in the case of zero empirical error, for a more general setting in [12]. In this paper, we address this question for the k-partite setting. We start by defining in Section 2 notions of the empirical and expected error of a k-partite ranking function, and then present in Section 3 our generalization bounds for k-partite ranking algorithms.

2 *k*-Partite Ranking Error

The notions of empirical k-partite ranking error defined in Eqs. (5) and (6) can be generalized as follows:

Definition 1 (Empirical k-partite ranking error). *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$. Let $S = ((x_1, y_1), \ldots, (x_M, y_M)) \in (\mathcal{X} \times \mathcal{Y})^M$, and for each l, let n_l denote the number of examples in S with rating l. Define the* empirical k-partite ranking error *of f with respect to S, parametrized by $\alpha \geq 0$ and denoted by $\widehat{R}_\alpha(f;S)$, as*

$$\widehat{R}_\alpha(f;S) \;=\; \frac{1}{C(S_Y)} \sum_{l=1}^{k-1} \sum_{m=l+1}^{k} \sum_{\{i:y_i=l\}} \sum_{\{j:y_j=m\}} (m-l)^\alpha\, \lambda(f, x_j, x_i)\,.$$

For $k = 2$, $\widehat{R}_\alpha(f;S)$ reduces to the bipartite ranking error for each $\alpha \geq 0$. In order to define a notion of the expected k-partite ranking error of a ranking function, we need to introduce

[1] The problem considered in [8] actually lies between ordinal regression and k-partite ranking; the setting is similar to that of k-partite ranking described above, but the goal is to learn a function $h : \mathcal{X} \to \mathcal{Y}$ that assigns ratings to new instances, and the error of such a function is measured by

$$\widehat{Q}(h;S) \;=\; \frac{1}{C(S_Y)} \sum_{l=1}^{k-1} \sum_{m=l+1}^{k} \sum_{\{i:y_i=l\}} \sum_{\{j:y_j=m\}} \mathbf{I}_{\{h(x_j) \leq h(x_i)\}}\,.$$

some notation; our notation will follow largely that of [1]. As is standard in classification, we shall assume that all examples (x, y) are drawn randomly and independently according to some fixed (but unknown) distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. For each $l \in \{1, \ldots, k\}$, the notation $\mathcal{D}_l$ will be used to denote the 'class-conditional' distribution $\mathcal{D}_{x|y=l}$. Several of our results will involve the conditional distribution $\mathcal{D}_{S_X|S_Y=\mathbf{y}}$ for some label sequence $\mathbf{y} = (y_1, \ldots, y_M) \in \mathcal{Y}^M$; this distribution is simply $\mathcal{D}_{y_1} \times \ldots \times \mathcal{D}_{y_M}$.

We would like to define the expected k-partite ranking error of a ranking function $f : \mathcal{X} \to \mathbb{R}$ as its expected 'ranking loss' on a pair of instances drawn from distinct classes. It turns out that, unlike the bipartite case, the expected loss in the general k-partite case depends on the class probabilities. Thus, we define the expected error as follows:

Definition 2 (Expected k-partite ranking error). *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$. Let $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_k) \in [0, 1]^k$ be a 'class probability' vector satisfying $\sum_{l=1}^{k} \rho_l = 1$. Define the* expected k-partite ranking error *of f with respect to $\boldsymbol{\rho}$, parametrized by $\alpha \geq 0$ and denoted by $R_\alpha(f; \boldsymbol{\rho})$, as*

$$R_\alpha(f; \boldsymbol{\rho}) \;=\; \frac{1}{\gamma(\boldsymbol{\rho})} \sum_{l=1}^{k-1} \sum_{m=l+1}^{k} \rho_l \rho_m \mathbf{E}_{x \sim \mathcal{D}_l, x' \sim \mathcal{D}_m} \Big\{ (m-l)^\alpha \lambda(f, x', x) \Big\} ,$$

where $\gamma(\boldsymbol{\rho}) = \sum_{l=1}^{k-1} \sum_{m=l+1}^{k} \rho_l \rho_m$.

In an ideal situation, one would like to estimate the expected error of a learned ranking function w.r.t. the true class probabilities (under $\mathcal{D}$). This is difficult for two reasons: first, the true class probabilities are generally unknown; second, the true probabilities may be different from the empirical class probabilities. In this paper, we study how well one can estimate the expected error w.r.t. the *empirical* class probabilities.

Definition 3 (Skew vector). *Let $\mathbf{y} = (y_1, \ldots, y_M) \in \mathcal{Y}^M$ be a finite rating sequence of length $M \in \mathbb{N}$, and for each $l \in \{1, \ldots, k\}$, let $n_l = |\{i : y_i = l\}|$. Define the* skew vector *of $\mathbf{y}$, denoted by $\boldsymbol{\rho}(\mathbf{y})$, as*

$$\boldsymbol{\rho}(\mathbf{y}) \;=\; (n_1/M, \ldots, n_l/M) .$$

For each l, we shall denote by $\rho_l(\mathbf{y})$ the lth component of $\boldsymbol{\rho}(\mathbf{y})$, i.e., $\rho_l(\mathbf{y}) = n_l/M$.

We shall be interested in bounding the deviation of the empirical error of a learned ranking function f_S, w.r.t. the training sample S from which it is learned, from the expected error of f_S with respect to the skew vector of S_Y.

3 Generalization Bounds

We give here two generalization bounds for k-partite ranking algorithms. The first bound applies to algorithms that select a ranking function from a finite function class; the second bound can be applied to algorithms that search potentially infinite function classes. Note that our results are all distribution-free, in the sense that they hold for any distribution $\mathcal{D}$.

Theorem 1. *Let $\mathcal{F}$ be a finite class of real-valued functions on $\mathcal{X}$, and let $\mathcal{A}$ be a k-partite ranking algorithm that, given a training sample $S \in (\mathcal{X} \times \mathcal{Y})^M$, returns a ranking function $f_S \in \mathcal{F}$. Let $\alpha \geq 0$. Then for any $0 < \delta \leq 1$, with probability at least $1 - \delta$ (over the draw of S according to $\mathcal{D}^M$), we have*

$$\left| \widehat{R}_\alpha(f_S; S) - R_\alpha(f_S; \boldsymbol{\rho}(S_Y)) \right| \;<\; \sqrt{\left(\frac{\ln |\mathcal{F}| + \ln\left(\frac{2}{\delta}\right)}{2M} \right) \sum_{l=1}^{k} \rho_l(S_Y) \left(C_{l,\alpha}(S_Y) \right)^2} ,$$

where for $\mathbf{y} \in \mathcal{Y}^M$, $C_{l,\alpha}(\mathbf{y}) = \Big(\sum_{m \neq l} |m-l|^\alpha \rho_m(\mathbf{y}) \Big) \Big/ \gamma(\boldsymbol{\rho}(\mathbf{y}))$.

The proof of the above bound relies on the derivation of a large deviation result for the k-partite ranking error, which in turn relies on a powerful concentration inequality of McDiarmid [9] and is similar to the derivation of such a result for the bipartite case [1]. Lack of space precludes a full discussion here; complete details of the proof can be found in [10].

Our second generalization bound, which is applicable to k-partite ranking algorithms that select a ranking function from a potentially infinite function class, is expressed in terms of a new set of combinatorial parameters that we term the *k-partite rank-shatter coefficients*. These coefficients extend naturally the bipartite rank-shatter coefficients of [1].

Definition 4 (Bipartite rank matrix [1]). *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$, let $m, n \in \mathbb{N}$, and let $\mathbf{x} = (x_1, \ldots, x_m) \in \mathcal{X}^m$, $\mathbf{x}' = (x'_1, \ldots, x'_n) \in \mathcal{X}^n$. The* bipartite rank matrix *of f with respect to $(\mathbf{x}, \mathbf{x}')$, denoted by $\mathbf{B}_f(\mathbf{x}, \mathbf{x}')$, is defined to be the matrix in $\{0, 1/2, 1\}^{m \times n}$ whose (i, j)-th element (for $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$) is given by*

$$[\mathbf{B}_f(\mathbf{x}, \mathbf{x}')]_{ij} \quad = \quad \mathbf{I}_{\{f(x_i) > f(x'_j)\}} + \frac{1}{2}\mathbf{I}_{\{f(x_i) = f(x'_j)\}} \,.$$

Definition 5 (k-Partite rank matrix set). *Let $f : \mathcal{X} \to \mathbb{R}$ be a ranking function on $\mathcal{X}$, let $n_1, \ldots, n_k \in \mathbb{N}$, and for each $l \in \{1, \ldots, l\}$, let $\mathbf{x}^{(l)} = (x_1^{(l)}, \ldots, x_{n_l}^{(l)}) \in \mathcal{X}^{n_l}$. Define the k-*partite rank matrix set *of f with respect to $(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)})$, denoted by $\mathbf{K}_f(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)})$, to be the following set of $\binom{k}{2}$ bipartite rank matrices:*

$$\mathbf{K}_f(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}) \quad = \quad \left\{ \mathbf{B}_f(\mathbf{x}^{(l)}, \mathbf{x}^{(m)}) \,\middle|\, 1 \leq l < m \leq k \right\}.$$

Definition 6 (k-Partite rank-shatter coefficient). *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $n_1, \ldots, n_k \in \mathbb{N}$. Define the $(n_1, \ldots, n_k)$-*th k-partite rank-shatter coefficient *of $\mathcal{F}$, denoted by $r(\mathcal{F}, n_1, \ldots, n_k)$, as follows:*

$$r(\mathcal{F}, n_1, \ldots, n_k) \quad = \quad \max_{\mathbf{x}^{(l)} \in \mathcal{X}^{n_l}} \left| \left\{ \mathbf{K}_f(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}) \mid f \in \mathcal{F} \right\} \right|.$$

Clearly, for finite $\mathcal{F}$, we have $r(\mathcal{F}, n_1, \ldots, n_k) \leq |\mathcal{F}|$ for all $n_1, \ldots, n_k$. In general, $r(\mathcal{F}, n_1, \ldots, n_k) \leq \prod_{l=1}^{k-1} \prod_{m=l+1}^{k} 3^{n_l n_m}$ for all $n_1, \ldots, n_k$.

Theorem 2. *Let $\mathcal{F}$ be a class of real-valued functions on $\mathcal{X}$, and let $\mathcal{A}$ be a k-partite ranking algorithm that, given a training sample $S \in (\mathcal{X} \times \mathcal{Y})^M$, returns a ranking function $f_S \in \mathcal{F}$. Let $\alpha \geq 0$. Then for any $0 < \delta \leq 1$, with probability at least $1 - \delta$ (over the draw of S according to $\mathcal{D}^M$), we have*

$$\left| \widehat{R}_\alpha(f_S; S) - R_\alpha(f_S; \boldsymbol{\rho}(S_Y)) \right|$$

$$< \quad \sqrt{8 \left(\frac{\ln r\big(\mathcal{F}, 2M\rho_1(S_Y), \ldots, 2M\rho_k(S_Y)\big) + \ln\left(\frac{4}{\delta}\right)}{M} \right) \sum_{l=1}^{k} \rho_l(S_Y) \left(C_{l,\alpha}(S_Y)\right)^2} \,,$$

where $C_{l,\alpha}(S_Y)$ is as defined in Theorem 1.

The proof of the above bound relies on the derivation of a uniform convergence result for the k-partite ranking error, and makes use of techniques similar to those used for the bipartite case [1]. Details (excluded here due to lack of space) can be found in [10].

The above bound is meaningful only if $r(\mathcal{F}, 2n_1, \ldots, 2n_k)$ grows sufficiently slowly with $n_1, \ldots, n_k$. Using properties of the bipartite rank-shatter coefficients [1], it can be shown that this is the case for certain function classes $\mathcal{F}$. In particular, one can obtain the following polynomial upper bound on the k-partite rank-shatter coefficients for linear ranking functions (see [10] for details):

Theorem 3. *For $d \in \mathbb{N}$, let $\mathcal{F}_{\text{lin}(d)}$ denote the class of linear ranking functions on $\mathbb{R}^d$. Then for all $n_1, \ldots, n_k \in \mathbb{N}$,*

$$r(\mathcal{F}_{\text{lin}(d)}, n_1, \ldots, n_k) \;\leq\; \prod_{l=1}^{k-1} \prod_{m=l+1}^{k} \left(\frac{2en_l n_m}{d} \right)^d .$$

A similar result can be shown for higher-order polynomial ranking functions.

4 Conclusion

The k-partite setting of the ranking problem is a natural setting that encompasses a wide range of applications. Our goal in this paper has been to initiate a formal study of this setting; we have defined notions of k-partite ranking error suitable for measuring the quality of ranking functions in the k-partite setting, and have obtained generalization bounds for k-partite ranking algorithms. While the basic techniques used to derive our results are similar to those used in the bipartite setting (which constitutes a special case of the more general k-partite setting), there are several important differences, including in particular the need in the k-partite case to define the expected error of a ranking function with respect to a vector $\boldsymbol{\rho}$ of 'class probabilities', and to consider convergence of the empirical error to the expected error with respect to an appropriate vector. It remains an open problem to derive bounds on the expected error with respect to the unknown vector of true class probabilities. Indeed, our results should be viewed as a first step in understanding the generalization behaviour of k-partite ranking algorithms. We expect that they will open the door to further analyses. For example, it can be shown that the (empirical) k-partite ranking error can be expressed as a U-statistic, which suggests that it may be possible to use tools of U-statistics [4].

Acknowledgments

We would like to thank Thore Graepel and Ralf Herbrich for pointing us to the k-partite generalization of the bipartite ranking problem. We would also like to express our gratitude to an anonymous reviewer for many insightful comments and suggestions.

References

[1] S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth. Generalization bounds for the area under the ROC curve. *Journal of Machine Learning Research*, 6:393–425, 2005.

[2] S. Agarwal and P. Niyogi. Stability and generalization of bipartite ranking algorithms. In *Proceedings of the 18th Annual Conference on Learning Theory*, 2005.

[3] S. Agarwal and D. Roth. Learnability of bipartite ranking functions. In *Proceedings of the 18th Annual Conference on Learning Theory*, 2005.

[4] S. Clémençon, G. Lugosi, and N. Vayatis. Ranking and scoring using empirical risk minimization. In *Proceedings of the 18th Annual Conference on Learning Theory*, 2005.

[5] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[6] K. Crammer and Y. Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, 2002.

[7] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[8] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132, 2000.

[9] C. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics 1989*, pages 148–188. Cambridge University Press, 1989.

[10] S. Rajaram and S. Agarwal. Generalization bounds for k-partite ranking. Report, 2005. Available from `http://www.ifp.uiuc.edu/~rajaram1/k-ranking-report.pdf`.

[11] S. Rajaram, A. Garg, X. S. Zhou, and T. S. Huang. Classification approach towards ranking and sorting problems. In *Proc. of the 14th European Conference on Machine Learning*, 2003.

[12] C. Rudin, C. Cortes, M. Mohri, and R. E. Schapire. Margin-based ranking meets boosting in the middle. In *Proceedings of the 18th Annual Conference on Learning Theory*, 2005.

Ranking with unlabeled Data: A first study

Nicolas Usunier, Vinh Truong, Massih R. Amini, Patrick Gallinari
University Pierre and Marie Curie
Computer Science Department
75015 Paris, France
{usunier, truong, amini, gallinari}@poleia.lip6.fr

Abstract

In this paper, we present a general learning framework which treats the ranking problem for various Information Retrieval tasks. We extend the training set generalization error bound proposed by [4] to the ranking case and show that the use of unlabeled data can be beneficial for learning a ranking function. We finally discuss open issues regarding the use of the unlabeled data during training a ranking function.

1 Introduction

Many learning applications are concerned with the ranking of objects from a fixed collection $\mathcal{C}$ upon a given query. This is the case, for example, in document retrieval or metasearch where the goal is to rank documents from a collection (i.e. the Web or intranets) based on their relevancy to a user's query. Another example is the automatic text summarization task seen as the 7extraction of relevant text-spans (sentences or paragraphs), where the user provides a document and the system returns a ranked list of text-spans from that document where the top ranked spans are supposed to reflect most, the content of the document. Although there are an increasing interest for the application of Machine Learning algorithms in these domains, the proposed learning settings lack a clear statistical framework.

In the other hand, most computational models proposed for ranking rely only on labeled training examples and ignore the possible information brought from unlabeled data. While a reasonable accuracy may be reached after training such a system on a large set of labeled data, labeling large amounts of data for learning may require expensive human resources and is often unrealistic. In this paper we propose a general setting for learning scoring functions for ranking which handles the additionnal information provided by unlabeled data for learning. The aim is to show that unlabeled data can help the learning process. Up to our knowledge this is the first time that the use of unlabeled data for ranking has been considered.

This paper is organized as follows; we first show how the Information Retrieval applications we consider could be handled by the proposed approach (section 2) and then present in section 3, the extension of the cross-validation bound of [4] to the ranking.

2 Ranking in Information Retrieval tasks

We consider Information Retrieval (IR) tasks where the system gets a user query $x \in \mathcal{X}$ and must return a subset $\mathcal{C}_x$ of a given collection $\mathcal{C}$, ordered in such a way that the first elements presented to the user should be the more relevant to his/her query. We consider the problem of learning a function that ranks the elements of $\mathcal{C}_x$ given x. If we furthermore assume **(1)** that there exist a fixed indexing of all subsets $\mathcal{C}_x$ of the form $\mathcal{C}_x = \{z^0, ..., z^{N_x-1}\}$, where N_x is the number of elements in $\mathcal{C}_x$, and **(2)** that for all x, $N_x \leq N$, the problem can be expressed as learning a function $\bar{f} : \mathcal{X} \mapsto \sigma_N$, where σ_N is the set of all permutations of $\{0, ..., N-1\}$, and the ranked list of the elements of $\mathcal{C}_x$ is given by $[z^{\bar{f}(x)^{-1}(i)}]_{i=0..N_x-1}$, where $\bar{f}(x)^{-1}$ is the inverse of $\bar{f}(x)$[1].

When learning $\bar{f}$ in a supervised setting, the training set $\mathcal{S}$ generally takes the form of n queries x_i, together with vectors $y_i = (y_i^k), y_i^k \in \{-1, 1\}, 0 \leq k \leq N_{x_i} - 1$, where y_i^k corresponds to a binary relevance judgement of $z_i^k \in \mathcal{C}_{x_i}$. Although this may not include all IR tasks, we restrict ourselves to this kind of supervision in the paper, since it greatly simplifies the presentation and encompasses most practical cases. Denoting $\mathcal{Y} = \bigcup_{k \leq N} \{-1, 1\}^k$, and $\Delta : \mathcal{Y} \times \sigma_N \mapsto \mathcal{R}_+$ being a risk function, we can define in a standard way the quantities describing the learning task. Given a fixed but unknown distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$, the true (or generalization) risk of $\bar{f}$ is $\epsilon(\bar{f}) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \Delta(y, \bar{f}(x))$, which can be estimated on $\mathcal{S} = (x_i, y_i)_{i=1}^n$ (assuming the (x_i, y_i)s are drawn i.i.d. according to $\mathcal{D}$) by the empirical risk $\epsilon(\bar{f}, \mathcal{S}) = \frac{1}{n} \sum_{i=1}^n \Delta(y_i, \bar{f}(x_i))$.

We will be interested in estimating $\epsilon(\bar{f})$, where $\bar{f}$ is chosen based on its training set $\mathcal{S}$, using a set of *unlabeled data* $\mathcal{S}_u = (x'_j)_{j=1}^m$, assumed to be drawn i.i.d. according to $\mathcal{D}_\mathcal{X}$, the marginal of $\mathcal{D}$ on $\mathcal{X}$ (notice that together with the x'_js, the learner has access to $\mathcal{C}_{x'_j}$ even for the unlabeled example). As we will see, such bounds are highly related to the loss function used, which we fix to the following for its convenience in the analysis and in practical applications $\Delta(y, \bar{f}(x)) = \frac{1}{p_x q_x} \sum_{i:y^i=1} \sum_{j:y^j=-1} [[\bar{f}(x)(i) > \bar{f}(x)(j)]]$ where p_x is the number of elements z^k of $\mathcal{C}_x$ with $y^k = 1$ and $q_x = N_x - p_x$, and $[[pr]]$ is one if predicate pr holds and zero otherwise.

Practical issues In practice, a convenient way to learn the function $\bar{f}$ is to learn a scoring function $f : \mathcal{Z} \subset \mathbb{R}^d \mapsto \mathbb{R}$, using a mapping $\Phi : \mathcal{X} \times \mathcal{C} \mapsto \mathcal{Z}$, which is a joint representation of x and an element of $\mathcal{C}_x$. Given an input x, $\bar{f}(x)$ can be induced from f such that its restriction to $\{1, ..., N_x - 1\}$ is the permutation of $\{0, ..., N_x - 1\}$ which satisfies $\bar{f}(x)(i) < \bar{f}(x)(j) \Leftrightarrow f(\Phi(x, z^i)) > f(\phi(x, z^j))$ (ties are broken arbitrarily), and then setting $\bar{f}(x)(i) = i$ for $i \geq N_x$. For example, in metasearch, $\mathcal{C}$ is the document collection, x a query, $\mathcal{C}_x$ a predefined set of documents extracted from the various search engines, and, for $z \in \mathcal{C}_x$, $\Phi(x, z)$ can be a vector where each dimension is the score returned by a given search engine, and we learn a scoring function f which is a combination of the scores. In extractive text summarization, $\mathcal{C}$ would be the set of all sentences appearing in a document collection, x a document, $\mathcal{C}_x$ the set of sentences appearing in document x, and $\Phi(x, z)$ is a vector of various heuristic scores, each one measuring how much the sentence z satisfies some criterion indicative of wether z reflects the content of x or not.

In terms of optimization, one can learn for example a discriminative linear function of the form $f(\Phi(x, z)) = w.\Phi(x, z)$ with $w \in \mathbb{R}^d$. If we consider, for a given x, two elements z^i and z^j of $\mathcal{C}_x$ such that $y^i = 1$ and $y^j = -1$, we have that $[[\bar{f}(x)(i) > \bar{f}(x)(j)]] = [[w.(\Phi(x, z^j) - \Phi(x, z^i)) > 0]]$, and the empirical risk with the loss Δ resembles a pairwise classification loss of all such pairs z^i, z^j on the training set. Standard classification algorithms can then be adapted to minimize the empirical risk defined in the previous sec-

[1] When $N_x < N$, we restrict $\bar{f}(x)(i) = i$ for $N_x \leq i \leq N - 1$.

tion. An example of such an optimization for the case of extractive text summarization can be found in [1]. By the same argument, even an algorithm like RankBoost [2] has to be adapted to be used in information retrieval in order to take into account this difference with the standard pairwise classification setting for ranking.

3 cross-validation bounds for ranking in IR tasks

A possible use of unlabeled data for classification has been introduced by [4]. The main idea of his work is to define a notion of distance between two classifiers c_1 and c_2 that can be computed only based on the unlabeled examples. Basically, if we consider a classification task where examples are in a space $\mathcal{X}$, have their labels in some space $\mathcal{Y}$, and if we have a classification loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathtt{R}_+$ such that ℓ verifies the triangle inequality (e.g. the $0-1$ loss), we have that $\forall (x,y) \in \mathcal{X} \times \mathcal{Y}, \ell(c_1(x), y) \leq \ell(c_2(x), y) + \ell(c_1(x), c_2(x))$. Under the standard assumption that the data (x, y) are drawn according to some probability distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ and denoting $\mathcal{D}_{\mathcal{X}}$ its marginal on $\mathcal{X}$, the quantity $\mathbb{E}_{x \sim \mathcal{D}_{\mathcal{X}}} \ell(c_1(x), c_2(x))$ bounds the differrence of generalization error between c_1 and c_2 and can be estimated using a set of unlabeled examples.

If we want to borrow the same idea in our case of ranking, we have to face the fundamental difference between our framework and classification, which is that the output of the function we learn is not in the same space as the labels associated to the examples. The function Δ defines a loss between a partial ranking (relevant elements must be presented before irrelevant ones), while the function outputs a total ranking over the elements. As a consequence, there is even no way of defining a triangle inequality. However, the core of the idea of [4] is kept if we can define a quantity $D(\bar{f}, \bar{f}')$, which can be estimated on an unlabeled sample given two ranking functions $\bar{f}$ and $\bar{f}'$, and which satisfies $\epsilon(\bar{f}) \leq \epsilon(\bar{f}') + D(\bar{f}, \bar{f}')$. The next subsection aims at defining such a D, while the following one follows the same steps as [Matti] to show a cross-validation bound for our cases of ranking.

definition of D We consider here a fixed example x, together with its label $y = (y^1, ..., y^K)$ where $K = N_x$, and two fixed ranking functions $\bar{f}$ and $\bar{f}'$. We will denote by p the number of y^is with value 1 and q the number of them of value -1. We have then $\Delta(y, \bar{f}(x)) = \frac{1}{pq} \sum_{i:y^i=1} \sum_{j:y^j=-1} [[\bar{f}(x)(i) > \bar{f}(x)(j)]]$. We will furthermore use the following additional notations: $rg(i) = \bar{f}(x)(i)$, whiwh represents the rank (starting from 0) of the element z^i of $\mathcal{C}_x$ in the ranked list output by $\bar{f}$, and, for i such that $y^i = 1$, $rg_+(i) = \sum_{k:y_k=1} [[rg(k) < rg(i)]]$, which counts the number of relevant elements ranked before the relevant element z^i. We define in the same way the quantities $rg'(i)$ and $rg'_+(i)$ for $\bar{f}'$.

Since our definitions of ranks start at zero, we have that $\Delta(y, \bar{f}(x)) = \frac{1}{pq} \sum_{i:y^i=1} (rg(i) - rg_+(i))$. Furthermore, we can notice that $\sum_{i:y^i=1} rg_+(i)$ only depends on p (and not on $\bar{f}$, since it represents the relative rankings of the relevant elements only. As a consequence, we can notice that $\Delta(y, \bar{f}(x)) - \Delta(y, \bar{f}'(x)) = \frac{1}{pq} \sum_{i:y^i=1} (rg(i) - rg'(i))$. Denoting by $\delta(\bar{f}(x), \bar{f}'(x))$ the list of length K containing all the values of $rg(i) - rg'(i)$ for $1 \leq i \leq K$ ordered in decreasing value, and letting $\delta(\bar{f}(x), \bar{f}'(x))_k$ the k-th element of the list (i.e. the k-th greatest value of $rg(i) - rg'(i)$ for $i \in \{1, ..., K\}$), it is easy to check that:

$$\Delta(y, \bar{f}(x)) - \Delta(y, \bar{f}'(x)) \leq \max_{p,q:p+q=K} \frac{1}{pq} \sum_{k=1}^{p} \delta(\bar{f}(x), \bar{f}'(x))_k \tag{1}$$

Finally denoting by $d(\bar{f}(x), \bar{f}'(x)) = \max_{p,q:p+q=N_x} \frac{1}{pq} \sum_{k=1}^{p} \delta(\bar{f}(x), \bar{f}'(x))_k$ and $D(\bar{f}, \bar{f}') = \mathbb{E}_{x \sim \mathcal{D}_{\mathcal{X}}} d(\bar{f}(x), \bar{f}'(x))$, we have that $D(\bar{f}, \bar{f}')$ can be estimated using a set

of unlabeled examples, and, taking the expectation over (x, y) drawn according to $\mathcal{D}$ of equation (1), we have $\epsilon(\bar{f}) \leq \epsilon(\bar{f}') + D(\bar{f}, \bar{f}')$.

A generalization error bound based on cross-validation for ranking[2] Like in classification, we define a randomized ranking function as being a σ_N-valued random variable that may depend on the input x but is independent from other randomized ranking function. In our work, we will only consider randomized ranking functions $\bar{f}_\Theta$ defined by a finite set of ranking functions $\{\bar{f}_k : \mathcal{X} \mapsto \sigma_N, k = 1, ..., K\}$ and a $\{1, ..., K\}$-valued random variable Θ, independent of all other random variables considered. When given an input instance x, a randomized ranking function choses a value $\theta \in \{1, ..., K\}$ according to he distribution Θ, and outputs $\bar{f}_\theta(x)$. When given a set of instances $x_i, i = 1, ..., n$, a value θ_i is drawn for each x_i, and a new copy of Θ is used each time, such that all the θ_i are independent of each other. Of course, a deterministic ranking function $\bar{f}$ as considered in the previous sections in the paper is a particular radomized ranking function, where the set contains only one element $\bar{f}_1 = \bar{f}$ and the random variable $\Theta = 1$ with probability 1. The notion of true risk extends to randomized ranking functions by setting $\epsilon(\bar{f}_\Theta) = \mathbb{E}_{\theta\sim\Theta}\mathbb{E}_{(x,y)\sim\mathcal{D}}\Delta(y, \bar{f}_\theta)$. Given two randomized ranking functions $\bar{f}_\Theta$ and $\bar{f}'_\Lambda$, the function D defined previously also extends, by setting $D(\bar{f}_\Theta, \bar{f}'_\Lambda) = \mathbb{E}_{\theta\sim\Theta,\lambda\sim\Lambda}\mathbb{E}_{x\sim\mathcal{D}_\mathcal{X}} d(\bar{f}_\theta(x), \bar{f}'_\lambda(x))$. Notice that given a realization θ of Θ and λ of Λ, we have $\Delta(y, \bar{f}_\theta(x)) - \Delta(y, \bar{f}'_\lambda(x)) \leq d(\bar{f}_\theta(x), \bar{f}'_\lambda(x))$, and therefore $\epsilon(\bar{f}_\Theta) \leq \epsilon(\bar{f}'_\Lambda(x)) + D(\bar{f}_\Theta, \bar{f}'_\Lambda)$.

Now consider a deterministic ranking function $\bar{f}$ that we just learned, and suppose there is a randomized ranking function $\bar{f}'_\Theta$, for which we can compute a tight bound $\hat{\epsilon}(\bar{f}'_\Theta, \delta/2)$ on $\epsilon(\bar{f}'_\Theta)$ which holds with probability at least $1 - \delta/2$ on the quantities needed to obtain the bound (e.g. the choices of the test set) and such that $D(\bar{f}, \bar{f}'_\Theta)$ may be small. The main idea is that if we can estimate an upper bound on $D(\bar{f}, \bar{f}'_\Theta)$, we will have a tight bound for $\bar{f}$. Then, assume we have a set $S_u = (x'_j)_{j=1}^m$ of unlabeled examples drawn i.i.d. according to $\mathcal{D}_\mathcal{X}$. Then, if we consider a vector $\theta = (\theta_1, ..., \theta_m)$ drawn according to Θ^m, $\frac{1}{m}\sum_{j=1}^m d(\bar{f}(x), \bar{f}'_{\theta_i}(x))$ is an unbiased estimate of $D(\bar{f}, \bar{f}'_\Theta)$ that can easily be computed. One can then use a large deviation inequality, like McDiarmid's theorem for bounded differences [5] (since $d(\bar{f}(x), \bar{f}'_{\theta_i}(x)) \in [0, 1]$), to obtain a computable upper bound $\hat{D}(\bar{f}, \bar{f}'_\Theta, S_u, \delta/2)$ on $D(\bar{f}, \bar{f}'_\Theta)$ which holds with probability $1 - \delta/2$ over the possibles S_u and $\theta = (\theta_1, ..., \theta_m)$. Then, with probability at least $1 - \delta$, we have:

$$\epsilon(\bar{f}) \leq \hat{\epsilon}(\bar{f}'_\Theta, \delta/2) + \hat{D}(\bar{f}, \bar{f}'_\Theta, S_u, \delta/2) \tag{2}$$

Finalizing the adaptation of [4] to our case, an interesting candidate for $\bar{f}'_\Theta$ is the randomized ranking function obtained by cross-validation $\bar{f}^{cv}_\Theta$ defined as follows. Given a labeled training set $S_l = (x_i, y_i)_{i=1}^n$, used to learn $\bar{f}$ with some algorithm, we arbitrarily split it into K disjoint parts $S_l^k, k = 1, ..., K$, each of size $n_k = \lfloor \frac{1}{K} \rfloor$ or $n_k = \lfloor \frac{1}{K} \rfloor + 1$ (such that $\sum_k n_k = n$), and, using the same algorithm as for $\bar{f}$, we train the ranking functions $\bar{f}_k$ on $\bigcup_{k' \neq k} S_l^{k'}$. $\bar{f}^{cv}_\Theta$ is then the randomized ranking function having $\{\bar{f}_k, k = 1, ..., K\}$ as set of ranking functions, and $\Theta = k$ with probability $\frac{1}{K}$ for $k = 1, ..., K$. The main interest of $\bar{f}^{cv}_\Theta$ is that since each $\bar{f}_k$ is trained using the same algorithm and a substantial part of the examples used to train $\bar{f}$, it is expected that on a given unlabeled set S_u, $\hat{D}(\bar{f}, \bar{f}^{cv}_\Theta, S_u, \delta/2)$ will be small. Furthermore, given $k \in \{1, ..., K\}$, $\bar{f}_k$ is independent of S_l^k, which can be used as a test set. McDiarmid's theorem can once again be used to obtain an upper bound $\hat{\epsilon}(\bar{f}_k, S_l^k, \frac{\delta}{2K})$ on $\epsilon(\bar{f}_k)$ based on its empirical estimate $\frac{1}{n_k}\sum_{x_i, y_i \in S_l^k} \Delta(y_i, \bar{f}_k(x_i))$, which holds with probability at least $1 - \frac{\delta}{2K}$ over the S_l^k. It is now easy to verify that

[2] This paragraph contains the straightforward extensions to the work of [4] needed to derive the bound.

$\hat{\epsilon}(\bar{f}_\Theta^{cv}, \delta/2) = \frac{1}{K}\sum_{k=1}^{K} \hat{\epsilon}(\bar{f}_k, S_l^k, \frac{\delta}{2K})$ is an upper bound on $\epsilon(\bar{f}_\Theta^{cv})$ which holds with probability $1 - \delta/2$ over the training sets S_l. Replacing $\bar{f}'_\Theta$ by $\bar{f}_\Theta^{cv}$ in equation (2) gives the desired computable generalization bound for $\bar{f}$ learned on S_l, using a set of unlabeled examples S_u.

4 Conclusion and Discussion

In this paper we showed some encouraging results by extending Kaaraiainen's work [4] to derive generalization error bounds based on cross-validation for a ranking function. We showed in particular that unlabeled data can be used with statistical guarantees for ranking in IR. However, the work in [4], going far beyond this specific bound, is mainly based on an embedding of the classifiers that can be learned and the target $\mathbb{P}(Y|X)$ in a pseudo-metric space of randomized classifiers. But the natural definition of randomized ranking functions is less convenient than randomized classifiers, since in the case of ranking, we do not have an equivalent for the target $\mathbb{P}(Y|X)$ that can be thought of as a randomized ranking function. The same kind of embedding becomes impossible, which shows that such natural extensions of classification results to ranking can not be done in a general way.

The framework we proposed in section 2, close to the label ranking framework proposed by [3], is largely sufficient to describe the supervised task of ranking in IR. However, it seems that specific analysis should be made for different loss functions in a semi-supervised setting. Due to the huge potential of unlabeled data for ranking in practical IR applications, we believe that it is a major issue to define a novel framework to deal with learning problems where the labeling of the examples is not in the domain of values of the function that we learn in which the semi-supervised ranking could be analyzed in a general way. We leave this as a direction for future work.

Acknowledgments

This work was supported by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors views.

References

[1] Amini M.R., Usunier N., Gallinari P., (2005) Automatic Text Summarization Based on Word Clusters and Ranking Algorithms, *Proceedings of the* 27^{th} *European Conference on Information Retrieval*.

[2] Dekel O., Manning C., and Singer Y., (2003) Log-Linear Models for Label Ranking. *NIP 2003*.

[3] Freund Y., Iyer R.D., Schapire R.E., Singer Y. (2003) An Efficient Boosting Algorithm for Combining Preferences, *Journal of Machine Learning Research 4*, pp. 933-969.

[4] Kääriäinen M. (2005) Generalization Error Bounds Using Unlabeled Data. *COLT'05*, pp. 127–142.

[5] McDiarmid C. (1989) On the method of bounded differences, *Surveys in Combinatorics*.

Extensions of Gaussian Processes for Ranking: Semi-supervised and Active Learning

Wei Chu
Gatsby Computational Neuroscience Unit
University College London
London, WC1N 3AR, UK
chuwei@gatsby.ucl.ac.uk

Zoubin Ghahramani
Gatsby Computational Neuroscience Unit
University College London
London, WC1N 3AR, UK
zoubin@gatsby.ucl.ac.uk

Abstract

Unlabelled examples in supervised learning tasks can be optimally exploited using semi-supervised methods and active learning. We focus on ranking learning from pairwise instance preference to discuss these important extensions, semi-supervised learning and active learning, in the probabilistic framework of Gaussian processes. Numerical experiments demonstrate the capacities of these techniques.

1 Introduction

Ranking learning is an important supervised problem of learning a ranking or ordering on instances, which has attracted considerable attention in machine learning research recently. Ranking learning, in which the training data are collected or interpreted in the form of pairwise instance preference, is also known as preference learning (Fürnkranz and Hüllermeier, 2005). These learning problems frequently arise in many applications, such as decision making, collaborative filtering and drug discovery. Various learning algorithms have been developed for preference learning, e.g. large margin classifiers (Herbrich et al., 1998; Aiolli and Sperduti, 2004), constraint classification approaches (Har-Peled et al., 2002), boosting-based learning algorithms (Dekel et al., 2004), neural networks with gradient descent methods (Burges et al., 2005) and probabilistic kernel approaches based on Gaussian processes (Chu and Ghahramani, 2005).

Many applications of ranking learning involve a large number of unlabelled examples and a few labelled examples, as expensive human effort is usually required in labelling examples. The issue of effectively exploiting the information in the unlabelled instances to facilitate supervised learning has been extensively studied under the name *semi-supervised learning* (Belkin et al., 2004; Chapelle et al., 2003; Lawrence and Jordan, 2005; Zhou et al., 2004; Zhu et al., 2003). Another issue, known as *active learning* (Cohn et al., 1995; Baram et al., 2004), concerns efficiently selecting queries from the unlabelled data pool sequentially to expedite the learning process.

This paper describes a fully probabilistic approach to active and semi-supervised ranking learning from pairwise instance preference using Gaussian processes. Inspired by the attractive concept of "warped RKHS" (Sindhwani et al., 2005b), semi-supervised Gaussian processes (Sindhwani et al., 2005a) have been recently developed which provide a general framework to incorporate unlabelled data for various supervised learning tasks and a princi-

pled way for model selection. We adapt this semi-supervised Gaussian process framework for ranking learning and further discuss active learning using information-theoretic criteria.

2 Pairwise Instance Preference

Consider a set of n distinct instances $x_i \in \mathcal{R}^d$ denoted as $\mathcal{X} = \{x_i : i = 1, \ldots, n\}$, and a set of K observed pairwise preference relations on the instances, denoted as

$$\mathcal{D} = \{v_k \succ u_k : k = 1, \ldots, K\} \tag{1}$$

where $v_k \in \mathcal{X}$, $u_k \in \mathcal{X}$, and $v_k \succ u_k$ means the instance v_k is ranked higher than u_k. For example, the pair $\{v_k, u_k\}$ could be two movies, while the user may prefer v_k to u_k. Note that only a small subset of $\mathcal{X}$ is labelled in $\mathcal{D}$.

Like previous approaches (Thurstone, 1927; Elo, 1978), we assume that each training sample x_i is associated with an unobservable latent function value $f(x_i)$, and these function values $\{f(x_i)\}$ preserve the preference relations in $\mathcal{D}$. We impose a Gaussian process prior on $\{f(x_i)\}$, and employ an appropriate likelihood function to learn the user's preference from the pairwise preferences between samples.

2.1 Semi-supervised Gaussian Processes

The latent function values $\{f(x_i)\}$ are assumed to be a realization of random variables in a zero-mean Gaussian process (Williams and Rasmussen, 1996). Traditionally the covariance between the latent functions corresponding to a pair of instances x_i and x_j can be defined by any reproducing kernel. A simple example is the Gaussian kernel defined as

$$\mathcal{K}(x_i, x_j) = \kappa_o \exp\left(-\frac{\kappa}{2}\sum_{\ell=1}^{d}(x_i^\ell - x_j^\ell)^2\right) \tag{2}$$

where $\kappa_o > 0$, $\kappa > 0$ and x_i^ℓ denotes the ℓ-th element of x_i. Thus the prior probability of these latent function values $\{f(x_i)\}$ is a multivariate Gaussian

$$\mathcal{P}(\boldsymbol{f}|\mathcal{X}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\boldsymbol{f}^T\Sigma^{-1}\boldsymbol{f}\right) \tag{3}$$

where $\boldsymbol{f} = [f(x_1), f(x_2), \ldots, f(x_n)]^T$, and Σ is the $n \times n$ covariance matrix whose ij-th element is the covariance function $\mathcal{K}(x_i, x_j)$ defined as in (2). Obviously this prior (3) does not contain any information of the abundant unlabelled data.

Semi-supervised Gaussian processes (Sindhwani et al., 2005a) provide a novel prior that exploits the localized spatial structure spanned by the given unlabelled data. A graph $\mathcal{G}$ can be defined over $\mathcal{X}$ by putting an edge between pairs of neighboring data points. For example, the neighbor set of each data point can be the k nearest neighbors simply. The resulting graph is denoted as $\mathcal{G} = (\mathcal{X}, \mathcal{E})$ where $\mathcal{E} \subset \mathcal{X} \times \mathcal{X}$ is the set of edges. Learning on the graph leads to the following (approximate) likelihood

$$\mathcal{P}(\mathcal{G}|\boldsymbol{f}) \propto \exp\left(-\frac{1}{2}\boldsymbol{f}^T L \boldsymbol{f}\right). \tag{4}$$

where L is defined to be the combinatorial Laplacian of the graph $\mathcal{G}$. For an unweighted graph the combinatorial Laplacian L is a sparse matrix where L_{ij} is equal to -1 (resp. 0) if $i \neq j$ and vertices x_i and x_j are connected (resp. disconnected), and $L_{ii} = -\sum_{j:j\neq i} L_{ij}$. The posterior distribution of $\boldsymbol{f}$ on the graph $\mathcal{G}$ is proportional to $\mathcal{P}(\mathcal{G}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})$, which is a multivariate Gaussian as follows

$$\mathcal{P}(\boldsymbol{f}|\mathcal{G}, \mathcal{X}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma^{-1} + L|^{-\frac{1}{2}}} \exp\left(-\frac{1}{2}\boldsymbol{f}^T(\Sigma^{-1} + L)\boldsymbol{f}\right). \tag{5}$$

The posterior distribution will be used as the prior distribution for the following supervised learning problem. Note that the new prior (5) is applicable to any data points in $\mathcal{R}^d$.

2.2 Likelihood Function and Posterior Distribution

The observation that v_k is preferred to u_k can be simply preserved by an inequality relation $f(v_k) > f(u_k)$. Using the similar ideas in the Thurstonian model (Thurstone, 1927) and Árpad Élő's Chess ranking system (Elo, 1978), we define a likelihood function for noisy observations on pairwise preference relations as follows,

$$\mathcal{P}\left(v_k \succ u_k | f_k\right) = \Phi\left(z_k\right) \tag{6}$$

where $f_k = [f(v_k), f(u_k)]^T$, $z_k = \frac{f(v_k)-f(u_k)}{\sqrt{2}\sigma}$ and $\Phi(z) = \int_{-\infty}^{z} \mathcal{N}(\gamma; 0, 1) d\gamma$ is the cumulative normal. Here $\sigma > 0$ reflects the noise level in measurement. Based on Bayes' theorem, the posterior probability can then be written as

$$\mathcal{P}(\boldsymbol{f}|\mathcal{D}) = \frac{\mathcal{P}(\boldsymbol{f})}{\mathcal{P}(\mathcal{D})}\mathcal{P}(\mathcal{D}|\boldsymbol{f}) = \frac{\mathcal{P}(\boldsymbol{f})}{\mathcal{P}(\mathcal{D})}\prod_{k=1}^{K}\mathcal{P}\left(v_k \succ u_k | f_k\right) \tag{7}$$

where the prior distribution $\mathcal{P}(\boldsymbol{f})$ is as defined in (5), the likelihood function is as defined in (6), and the normalization factor $\mathcal{P}(\mathcal{D}) = \int \mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})d\boldsymbol{f}$ is known as the evidence of the model parameters $\{\kappa_o, \kappa, \sigma\}$.

2.3 Expectation Propagation

Several approximate Bayesian inference methods can be applied to approximate the posterior distribution (7) as a Gaussian. In this work, we apply the expectation propagation (EP) algorithm (Minka, 2001; Csató, 2002). EP attempts to approximate $\mathcal{P}(\mathcal{D}|\boldsymbol{f})\mathcal{P}(\boldsymbol{f})$ as a parametric product distribution in the form $\mathcal{Q}(\boldsymbol{f}) = \mathcal{P}(\boldsymbol{f})\prod_{k=1}^{K}\tilde{t}(f_k) = \mathcal{P}(\boldsymbol{f})\prod_{k=1}^{K} s_k \exp\left(-\frac{1}{2}(f_k - m_k)^T \pi_k (f_k - m_k)\right)$, where $m_k = [m_{u_k}, m_{v_k}]^T$ and π_k is a 2×2 matrix. The parameters $\{s_k, m_k, \pi_k\}$ in $\{\tilde{t}(f_k)\}$ are successively optimized by minimizing the KL divergence,

$$\tilde{t}(f_k)^{\text{new}} = \arg\min_{\tilde{t}(f_k)} \mathbf{KL}\left(\frac{\mathcal{Q}(\boldsymbol{f})}{\tilde{t}(f_k)^{\text{old}}}\mathcal{P}(v_k \succ u_k | f_k) \,\middle\|\, \frac{\mathcal{Q}(\boldsymbol{f})}{\tilde{t}(f_k)^{\text{old}}}\tilde{t}(f_k)\right). \tag{8}$$

Since $\mathcal{Q}(\boldsymbol{f})$ is in the exponential family, this minimization can be simply solved by moment matching up to the second order. A detailed updating scheme can be found in Appendix A. The formulation is particularly useful in sequential learning, though there is no guarantee of convergence. At the equilibrium of $\mathcal{Q}(\boldsymbol{f})$, we obtain an approximate evidence defined as in (11) for model selection.

3 Active Learning

Learning could be made more efficient if we can actively select salient data points. Within the Bayesian learning framework, the expected informativeness of a new observation can be measured by the change in entropy of the posterior distribution of the latent functions by the inclusion of the candidate (MacKay, 1992; Lawrence et al., 2002). The new posterior distribution with the inclusion of the *unused* preference relation $(v_k \succ u_k)$ can be approximated as a Gaussian $\mathcal{N}(\boldsymbol{f}; \mathcal{A}^{new}, \boldsymbol{h}^{new})$ as described in Appendix A. The entropy gain can be evaluated by

$$\Delta\mathbf{H}_{v_k \succ u_k} = -\frac{1}{2}\log\det(\lambda_k^{new}\lambda_k^{-1}) \tag{9}$$

where λ_k and λ_k^{new} are as defined in Appendix A. For unlabelled preference relations, we can evaluate the expected entropy gain as the criterion,

$$\langle\Delta\mathbf{H}_k\rangle = \mathcal{P}(v_k \succ u_k|\mathcal{D})\Delta\mathbf{H}_{v_k \succ u_k} + \mathcal{P}(v_k \prec u_k|\mathcal{D})\Delta\mathbf{H}_{v_k \prec u_k} \tag{10}$$

where the predictive probability $\mathcal{P}(v_k \succ u_k|\mathcal{D})$ is evaluated by $\mathcal{Z}_k$ as in Appendix A. As a much more expensive alternative, the predictive distributions of all the unlabelled data can be updated by the inclusion of the candidate and then the sum of the expected entropy gain of the updated predictive distributions over all the unlabelled data could be used as the score. The strategy is to select the sample with the highest score from the data pool.

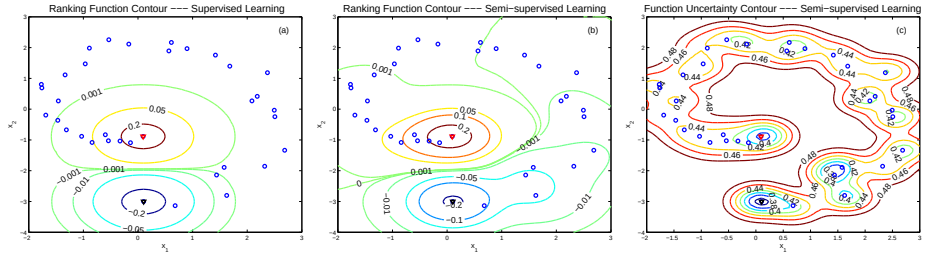

Figure 1: Contour graphs of the results on a synthetic spiral dataset. The dataset is represents by blue circles and the labelled pair is marked as triangles. The error bars of ranking functions in the posterior distribution are presented in graph (c).

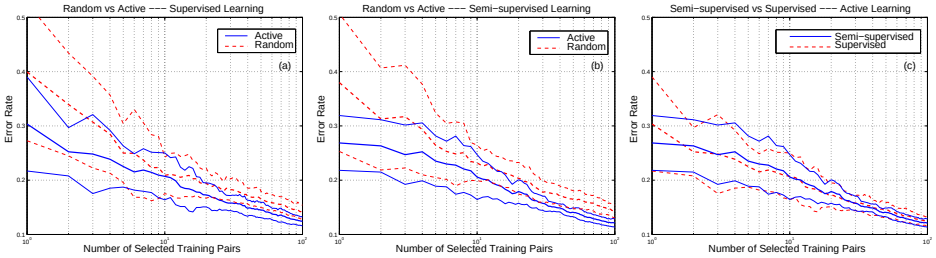

Figure 2: Performance of active learning on Boston Housing dataset. The error rate is the percent of incorrect preference predictions on 127765 pairs. The curves present the average over 20 trials along with standard deviation.

4 Demonstration

An artificial dataset was generated as shown in Figure 1, which is composed of 35 data points scattered along a spiral. Only one pair is labelled by preference. We used the Gaussian kernel defined as in (2) and the model parameters were set at $\kappa_o = 0.25$, $\kappa = 4.0$ and $\sigma^2 = 0.01$. The ranking function obtained by supervised learning is presented in Figure 1(a). We applied 3-nearest-neighbor to setup the graph $\mathcal{G}$ for semi-supervised learning and the corresponding results are presented in Figure 1(b-c). Clearly semi-supervised Gaussian processes have captured the density information that also significantly changed the ranking function and its error bars.

The Boston Housing dataset was used for active learning, which consists of 506 samples with 13 features.[1] The ranking over these samples was decided by the "housing price". We sequentially selected a pair of samples from the unlabelled data pool (1000 randomly selected unlabelled pairs) for query and then added the labelled pair into training. We used a Gaussian kernel and the model parameters were set at $\kappa_o = 0.0625$, $\kappa = 0.0625$ and $\sigma^2 = 0.001$. The graph for semi-supervised learning was constructed by 3-nearest-neighbor. We compared the performance of using the entropy criterion defined as in (10) against random selection in Figure 2(a-b). The entropy criterion works greatly better than random selection on both supervised and semi-supervised learning. We also compared active semi-supervised learning to active supervised learning in Figure 2(c). The active learning results are comparable on this case, whereas semi-supervised learning yields slightly better results than supervised learning after selecting 30 pairs.

Acknowledgment

This work was supported by the National Institutes of Health Grant Number 1 P01 GM63208. W. Chu thanks S. Sathiya Keerthi and Vikas Sindhwani for many discussions.

[1] The original data can be found in StatLib via http://lib.stat.cmu.edu/datasets/boston.

A EP Algorithm Outline

Let us define an augmented matrix Π_k for π_k, which is a $n \times n$ matrix with only four non-zero entries from π_k. Similarly we define a $n \times 1$ vector M_k for m_k. In notation of Π_k and M_k, we have $\mathcal{Q}(\boldsymbol{f}) = \mathcal{P}(\boldsymbol{f}) \prod_{k=1}^{K} s_k \exp\left(-\frac{1}{2}(\boldsymbol{f} - M_k)^T \Pi_k (\boldsymbol{f} - M_k)\right)$ equivalently, which is a Gaussian distribution $\mathcal{N}(\boldsymbol{f}; \boldsymbol{h}, \mathcal{A})$ with covariance $\mathcal{A} = (\Sigma^{-1} + L + \Pi)^{-1}$ and mean $\boldsymbol{h} = \mathcal{A}\Psi$ where $\Pi = \sum_{k=1}^{K} \Pi_k$ and $\Psi = \sum_{k=1}^{K} \Pi_k M_k$.

The initial states:

- site matrix $\pi_k = 0$, and site vector $\psi_k = \pi_k m_k = 0, \forall k$;
- site amplitude $s_k = 1$, posterior mean $\boldsymbol{h} = 0$, and posterior variance $\mathcal{A} = (\Sigma^{-1} + L)^{-1}$;

Looping k from 1 to K until there is no significant change in $\{\psi_k, \pi_k, s_k\}_{k=1}^{K}$:

- $\tilde{t}(f_k)^{old}$ is removed from $\mathcal{Q}(\boldsymbol{f})$, which leads to the leave-one-out distribution $\mathcal{Q}^{\backslash i}(\boldsymbol{f}) = \frac{\mathcal{Q}(\boldsymbol{f})}{\tilde{t}(f_k)^{old}}$ having (for *unused* preference relations $\lambda_k^{\backslash k} = \lambda_k$ and $h_k^{\backslash k} = h_k$)
  - covariance of f_k: $\lambda_k^{\backslash k} = (\lambda_k^{-1} - \pi_k)^{-1}$, where λ_k is a 2×2 sub-matrix of $\mathcal{A}$ with entries associated with the instances u_k and v_k;
  - mean of f_k: $h_k^{\backslash k} = h_k + \lambda_k^{\backslash k}(\pi_k h_k - \psi_k)$, where h_k is a 2×1 vector with entries of the posterior mean of $f(u_k)$ and $f(v_k)$;
- By incorporating $\mathcal{P}(v_k \succ u_k | f_k)$ into $\mathcal{Q}^{\backslash i}(\boldsymbol{f})$, the parameters in $\tilde{t}(f_k)$ can be computed as follows:
  - Let us denote $\mathbf{1}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and $\mathbf{1}_2 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$;
  - $\mathcal{Z}_k = \int \mathcal{P}(v_k \succ u_k | f_k) \mathcal{N}(f_k; h_k^{\backslash k}, \lambda_k^{\backslash k})\, df_k = \Phi(\tilde{z}_k)$, where $\tilde{z}_k = \frac{\mathbf{1}_1^T h_k^{\backslash k}}{\sigma_\star}$ and $\sigma_\star^2 = 2\sigma^2 + \text{trace}\left(\lambda_k^{\backslash k} \mathbf{1}_2\right)$;
  - $\frac{\partial \log \mathcal{Z}_i}{\partial h_k^{\backslash k}} = \gamma_k = \frac{\mathcal{N}(\tilde{z})}{\sigma_\star \Phi(\tilde{z})} \mathbf{1}_1$, and $\frac{\partial \log \mathcal{Z}_i}{\partial \lambda_k^{\backslash k}} = \beta_k = \frac{-\tilde{z}\mathcal{N}(\tilde{z})}{2\sigma_\star^2 \Phi(\tilde{z})} \mathbf{1}_2$;
  - $\omega_k = \gamma_k \gamma_k^T - 2\beta_k$;
  - $h_k^{new} = h_k^{\backslash k} + \lambda_k^{\backslash k} \gamma_k$;
  - $\lambda_k^{new} = \lambda_k^{\backslash k} - \lambda_k^{\backslash k} \omega_k \lambda_k^{\backslash k}$;
  - $\pi_k^{new} = (\lambda_k^{new})^{-1} - (\lambda_k^{\backslash k})^{-1}$;
  - $\psi_k^{new} = (\lambda_k^{new})^{-1} \lambda_k^{\backslash k} \gamma_k + \pi_k^{new} h_k^{\backslash k}$;
  - $B_k = h_k^{\backslash k T} (\lambda_k^{\backslash k})^{-1} h_k^{\backslash k} - (\psi_k^{new} + (\lambda_k^{\backslash k})^{-1} h_k^{\backslash k})^T \lambda_k^{new} (\psi_k^{new} + (\lambda_k^{\backslash k})^{-1} h_k^{\backslash k})$;
  - $s_k^{new} = \mathcal{Z}_k \left| \left((\lambda_k^{\backslash k})^{-1} + \pi_k^{new}\right) \lambda_k^{\backslash k} \right|^{\frac{1}{2}} \exp\left(\frac{1}{2} B_k\right)$; [2]
- update $\{\pi_k, \psi_k, s_k\}$, and update $\mathcal{A}$ and h as follows
  - $\mathcal{A}^{new} = \mathcal{A} + \mathbf{a}_k^T \Upsilon_k \mathbf{a}_k$, where $\Upsilon_k = \lambda_k^{-1}(\lambda_k^{new} - \lambda_k)\lambda_k^{-1}$ and $\mathbf{a}_k$ is a $2 \times n$ matrix containing the two columns of $\mathcal{A}$ associated with u_k and v_k;
  - $\boldsymbol{h}^{new} = \boldsymbol{h} + \eta \mathbf{a}_k$, where $\eta = \lambda_k^{-1} \lambda_k^{\backslash k} (\gamma_k + \pi_k h_k - \psi_k)$.

The approximate evidence at the equilibrium can be written as

$$\mathcal{P}(\mathcal{D}|\theta) \approx \frac{|\mathcal{A}|^{\frac{1}{2}}}{|\Sigma|^{\frac{1}{2}}} \exp\left(\frac{B}{2}\right) \prod_{k=1}^{K} s_k \tag{11}$$

where $B = \Psi^T \mathcal{A} \Psi$.

[2] The superfluous term $(m_k^{new})^T \pi_k^{new} m_k^{new}$ is removed from B_k.

References

Aiolli, F. and A. Sperduti. Learning preferences for multiclass problems. In *Advances in Neural Information Processing Systems 17*, 2004.

Baram, Y., R. E. Yaniv, and K. Luz. Online choice of active learning algorithms. *JMLR*, 5:255–291, 2004.

Belkin, M., P. Niyogi, and V. Sindhwani. Manifold regularization: a geometric framework for learning from examples. Computer Science Technical Report TR-2004-06, University of Chicago, 2004.

Burges, C., T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceeding of the 22th International Conference on Machine Learning*, pages 89–96, 2005.

Chapelle, O., J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *NIPS*, 2003.

Chu, W. and Z. Ghahramani. Preference learning with Gaussian processes. In *Proceeding of the 22th International Conference on Machine Learning*, pages 137–144, 2005.

Cohn, D. A., Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. In *Advances in Neural Information Processing Systems 7*, 1995.

Csató, L. *Gaussian Processes - Iterative Sparse Approximation*. Ph.D. thesis, Aston University, 2002.

Dekel, O., J. Keshet, and Y. Singer. Log-linear models for label ranking. In *Proceedings of the 21st International Conference on Machine Learning*, pages 209–216, 2004.

Elo, A. E. *The ratings of chess players: past and present*. London:Batsford, 1978.

Fürnkranz, J. and E. Hüllermeier. Preference learning. *Künstliche Intelligenz*, 2005. in press.

Har-Peled, S., D. Roth, and D. Zimak. Constraint classification: A new approach to multiclass classification and ranking. In *Advances in Neural Information Processing Systems 15*, 2002.

Herbrich, R., T. Graepel, P. Bollmann-Sdorra, and K. Obermayer. Learning preference relations for information retrieval. In *Proc. of Workshop Text Categorization and Machine Learning, ICML*, pages 80–84, 1998.

Lawrence, N. D. and M. I. Jordan. Semi-supervised learning via Gaussian processes. In *Advances in Neural Information Processing Systems 17*, pages 753–760, 2005.

Lawrence, N. D., M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In Becker, S., S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 609–616, 2002.

MacKay, D. J. C. Information-based objective functions for active data selection. *Neural Computation*, 4(4):589–603, 1992.

Minka, T. P. *A family of algorithms for approximate Bayesian inference*. Ph.D. thesis, Massachusetts Institute of Technology, January 2001.

Sindhwani, V., W. Chu, and S. S. Keerthi. Semi-supervised Gaussian processes. Technical report, Yahoo! Research, 2005a. in preparation.

Sindhwani, V., P. Niyogi, and M. Belkin. Beyound the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22th International Conference on Machine Learning*, pages 825–832, 2005b.

Thurstone, L. L. A law of comparative judgement. *Psychological Review*, 34:273–286, 1927.

Williams, C. K. I. and C. E. Rasmussen. Gaussian processes for regression. In Touretzky, D. S., M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 598–604, 1996. MIT Press.

Zhou, D., O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 18*, pages 321–328, 2004.

Zhu, X., Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceeding of the 20th International Conference on Machine Learning*, 2003.

Collaborative Ordinal Regression

Shipeng Yu[1,2], **Kai Yu**[2], **Volker Tresp**[2]
[1]Institute for Computer Science, University of Munich, Munich 80538, Germany
[2]Siemens Corporate Technology, Munich 81739, Germany
spyu@dbs.ifi.lmu.de
kai.yu@siemens.com, volker.tresp@siemens.com

Abstract

Ordinal regression has become an effective way of learning user preferences, but most of research only focuses on single regression problem. In this paper we introduce *collaborative ordinal regression*, where multiple ordinal regression tasks need to be handled simultaneously. Rather than modelling each task individually, we build a hierarchical Bayesian model and assign a common Gaussian Process (GP) prior to all individual latent functions. This very general model allows us to formally model the inter-dependencies between regression functions. We derive a very general learning scheme for this type of models, and in particular we evaluate two example models with collaborative effect. Empirical studies show that collaborative model outperforms the individual counterpart.

1 Introduction

Recent years have seen many works on preference learning, which aims to learn a preference model for the user and make predictions for newly arriving items. This is in general called ordinal regression in the literature [5]. In this paper we are interested in probabilistic conditional models for rankings (preference labels), for which we need to define some generative process for ranking data. Some recent work in this direction include [3] in which Chu and Ghahramani applied Gaussian Processes (GP) to ordinal regression, and [2] where Burges et al. derived a generative model for pairs of objects and trained the model using neural networks.

Up to now most of the research in ordinal regression focuses on a single regression problem, but in reality many ranking applications have multiple regression tasks. For instance, in user preference prediction, one user can be modelled as an ordinal regression problem, but in many cases we need to predict user preferences for many users at the same time. Intuitively, we should not model each user individually, but model all the users *jointly* to uncover the dependency between them. We call this problem *collaborative ordinal regression*, since it is more general than the well-known collaborative filtering problem. Another example of multiple regression is in web page ranking, where each query can be taken as an ordinal regression function (with possibly more than 2 labels) on all the web pages. The problem here is how to rank the web pages for a new query.

In this paper we propose a very general Bayesian framework for collaborative ordinal regression. The preference labels for one regression task are assumed to be generated from

a latent function, and all the latent functions share a common GP prior to account for the inter-dependencies. Learning in this model is via an EM-like algorithm, where the E-step is an Expectation-Propagation (EP) [6] iteration in the general case. The model is shown to cover many existing models in the literature, and we empirically compare two of them in a collaborative manner. The collaborative models are shown to give better performance than the individual models.

The rest of the paper is organized as follows. In Section 2 we formally introduce the collaborative framework and give some example models. Then in Section 3 we derive a learning algorithm for collaborative ordinal regression. Some empirical results are shown in Section 4, followed by Section 5 with some discussions and future directions.

2 Model Formulation

In this section we use the language of collaborative filtering to explain the model. We assume we have an item set of m items, with the ith item denoted as $\mathbf{x}_i \in \mathbb{R}^d$. Each user gives a preference label y to each item, which is an integer between 1 (lowest preference) and r (highest preference).

2.1 Ordinal Regression for Single User

In the single user case, the data consist of pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where y_i gives the preference label for item $\mathbf{x}_i$. One natural assumption is that there is an unobserved latent function $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$ which maps items into a real line, and the ranking outputs are then generated from the latent values $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m)$. More formally, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]^\top$, $\mathbf{y} = [y_1, \ldots, y_m]$ and $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m)]$, the likelihood of preference labels is written as

$$P(\mathbf{y}|\mathbf{X}, f, \theta) = P(\mathbf{y}|f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m), \theta) = P(\mathbf{y}|\mathbf{f}, \theta),$$

where θ is some model parameter.

To complete the Bayesian formulation, we assign a Gaussian prior to the function f, $f \sim \mathcal{N}(f; h, \mathbf{\Sigma})$, which leads to a GP model for preference learning (see [3] for more explanations). h is the *prior function* for f, and $\mathbf{\Sigma}$ is the *kernel matrix* which is generated from a *covariance function* $\kappa(\cdot, \cdot)$ (also known as *kernel function*). The (i, k)-th entry in the kernel $\mathbf{\Sigma}$ is calculated as $\kappa(\mathbf{x}_i, \mathbf{x}_k)$, which measures the covariance between the functions corresponding to inputs $\mathbf{x}_i$ and $\mathbf{x}_k$. The final conditional likelihood is written as

$$P(\mathbf{y}|\mathbf{X}, \theta, \phi) = \int P(\mathbf{y}|\mathbf{X}, f, \theta) P(f|\phi)\, df,$$

where $\phi \equiv \{h, \mathbf{\Sigma}\}$ denote all the parameters for prior of f.

2.2 Collaborative Ordinal Regression

In the multiple user case, we have n users, and user j has preference labels $\mathbf{y}_j$. Following the notation in previous subsection, we can model each user j as an ordinal regression model $P(\mathbf{y}_j|\mathbf{f}_j, \theta_j)$. However if we model them separately, label prediction for one user will only depend on preference labels of this particular user, which means we will lose the *collaborative effect* between users. To cope with this user dependency, we follow recent works for multi-task learning [7, 8] and assign a *common* Gaussian Process prior ϕ to all the functions f_j's in the GP model. In this nonparametric hierarchical Bayesian framework, different users are constrained to have similar preference patterns on the items. In particular, the prior function h defines the mean preference of these users, and kernel matrix $\mathbf{\Sigma}$ constrains the smoothness of these functions over all items. To better reflect the "common

interest" between users, both h and $\mathbf{\Sigma}$ can be adapted to the training data. The conditional likelihood for the whole observations $\mathcal{D} \equiv \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ is then written as

$$P(\mathcal{D}|\mathbf{X}, \mathbf{\Theta}, \phi) = \prod_{j=1}^{n} P(\mathbf{y}_j|\mathbf{X}, \theta_j, \phi) = \prod_{j=1}^{n} \int P(\mathbf{y}_j|\mathbf{X}, f_j, \theta_j) P(f_j|\phi)\, df_j. \quad (1)$$

To be more Bayesian, one can assign priors for θ_j's and for ϕ, which results in a more flexible yet more complicated model. In particular, we can assign a Normal-Inverse-Wishart prior for ϕ, as done in [7, 8]. For MAP estimate of ϕ, this corresponds to a smooth term in the learning process. In this paper we do not consider this prior for simplicity.

2.3 Example Models

In the literature, different ranking models differ in defining the likelihood term $P(\mathbf{y}|\mathbf{f}, \theta)$. In this paper we discuss and compare two models, in which the likelihood term can be factorized with respect to items, i.e., $P(\mathbf{y}|\mathbf{f}, \theta) = \prod_{i=1}^{m} P(y_i|f(\mathbf{x}_i), \theta)$. Other forms of likelihood is discussed briefly in Section 5.

Gaussian Process Regression (GPR) This model grants the problem simply as a GP regression problem. The ranking label y_i is assumed sampled from a Gaussian with mean $f(\mathbf{x}_i)$ and some variance σ^2, i.e., ($\theta \equiv \sigma^2$)

$$P(y_i|f(\mathbf{x}_i), \theta) = \mathcal{N}(y_i; f(\mathbf{x}_i), \sigma^2).$$

Gaussian Process Ordinal Regression (GPOR) This model is discussed in [3] and assumes there are some "pins" $b_0, b_1, \ldots, b_r$ on the real line. Each label y_i is assigned according to which area a disturbed value $z_i \sim \mathcal{N}(z_i; f(\mathbf{x}_i), \sigma^2)$ is in:

$$P(y_i|f(\mathbf{x}_i), \theta) = \int_{b_{y_i-1}}^{b_{y_i}} \mathcal{N}(z_i; f(\mathbf{x}_i), \sigma^2)\, dz_i = \Phi\left(\frac{b_{y_i} - f(\mathbf{x}_i)}{\sigma}\right) - \Phi\left(\frac{b_{y_i-1} - f(\mathbf{x}_i)}{\sigma}\right).$$

We can define $b_0 = -\infty$, $b_r = +\infty$, and the model parameter $\theta \equiv \{b_1, \ldots, b_{r-1}, \sigma\}$.

Both of these models can be generalized to collaborative models. In the following we call the collaborative versions of them as CGPR and CGPOR, respectively.

3 Learning

In this section we derive a very general framework for learning in collaborative ordinal regression model, where in principal we could use any likelihood function for $P(\mathbf{y}|\mathbf{f}, \theta)$. Since for collaborative filtering it is very likely to have missing data in the $\mathbf{y}$'s, we also consider this situation in the learning procedure. Starting from the likelihood function (1), we apply Jensen's inequality and obtain

$$\begin{aligned} \log P(\mathcal{D}|\mathbf{X}, \mathbf{\Theta}, \phi) &= \sum_{j=1}^{n} \log \int P(\mathbf{y}_j|\mathbf{X}, f_j, \theta_j) P(f_j|\phi)\, df_j \\ &\geq \sum_{j=1}^{n} \int Q(f_j) \log \frac{P(\mathbf{y}_j|\mathbf{X}, f_j, \theta_j) P(f_j|\phi)}{Q(f_j)}\, df_j, \end{aligned} \quad (2)$$

where $Q(f_j)$ is some function of f_j. Therefore an EM-like learning procedure can be derived by iteratively maximizing this lower bound (2) with respect to $Q(f_j)$ and model parameters $\mathbf{\Theta}$ and ϕ. In the E-step, we approximate $Q(f_j)$ to the *a posteriori* distribution of f_j, and we take a Gaussian form for Q, i.e., $Q(f_j) = \mathcal{N}(f_j; \hat{f}_j, \hat{\mathbf{\Sigma}}_j)$. While directly

maximizing the lower bound with respect to Q is difficult and may be intractable[1], we can apply Expectation-Propagation (EP) [6] to do this approximation, since $P(\mathbf{y}_j|\mathbf{X}, f_j, \theta_j)$ takes a factorized form for each j (see Appendix for details). This corresponds to Assumed Density Filtering (ADF) as an approximation to the posterior. In the M-step, new model parameters are obtained by directly maximizing the lower bound (2) with $Q(f_j)$ known from E-step. It is seen that θ_j's and ϕ are not coupled in this optimization problem. For θ_j we need to solve $\hat{\theta}_j = \arg\min_{\theta_j} \int Q(f_j) \log P(\mathbf{y}_j|\mathbf{X}, f_j, \theta)\, df_j$ analytically or numerically. The update for ϕ can be easily obtained as

$$\hat{h} = \frac{1}{n}\sum_j \hat{f}_j, \qquad \hat{\mathbf{\Sigma}} = \frac{1}{n}\sum_j \left[(\hat{f}_j - \hat{h})(\hat{f}_j - \hat{h})^\top + \hat{\mathbf{\Sigma}}_j\right], \tag{3}$$

which can be intuitively understood as an average over all the functions. We emphasize that this update is *independent* of the likelihood function form, and that different users are connected and influenced *only* via this update. Note here that we can learn both the mean function h and the (non-stationary) kernel matrix $\mathbf{\Sigma}$, because we have multiple samples of f_j's from the GP prior (see [8] for a detailed discussion). This is in contrast to model fitting in [3] where only a parameter for a stationary kernel is updated.

3.1 Learning in CGPR

In CGPR model, since the likelihood term $P(\mathbf{y}_j|\mathbf{X}, f_j, \theta_j)$ already takes a Gaussian form, the approximation in the E-step is exact. Denote m_j the length of $\mathbf{y}_j$, the E-step turns out to be

$$\hat{f}_j = \mathbf{\Sigma}_{m,j}(\mathbf{\Sigma}_{j,j} + \sigma^2\mathbf{I})^{-1}(\mathbf{y}_j - \mathbf{h}_j) + \mathbf{h}, \quad \hat{\mathbf{\Sigma}}_j = \mathbf{\Sigma} - \mathbf{\Sigma}_{n,j}(\mathbf{\Sigma}_{j,j} + \sigma^2\mathbf{I})^{-1}\mathbf{\Sigma}_{n,j}^\top,$$

where $\mathbf{h}_j$ is the length-m_j sub-vector of $\mathbf{h}$, and $\mathbf{\Sigma}_{m,j}$ and $\mathbf{\Sigma}_{j,j}$ denote the $m \times m_j$ and $m_j \times m_j$ sub-matrix of $\mathbf{\Sigma}$, respectively. The M-step for θ is given analytically as $\hat{\sigma}^2 = \frac{1}{\sum_j n_j}\sum_j \left(\|\mathbf{y}_j - \hat{f}_j\|^2 + \mathrm{tr}[\hat{\mathbf{\Sigma}}_j]\right)$. The whole algorithm turns out to be the same as in [7] and [8], if we assign a Normal-Inverse-Wishart hyperprior to ϕ and consider MAP estimates for ϕ.

3.2 Learning in CGPOR

The learning algorithm for CGPOR model is a natural extension of the EP algorithm given in [3]. In the E-step, we fit a Gaussian $Q(f_j)$ for the *a posteriori* distribution of each function f_j. Since there may be items without labels, we only need to consider the likelihood terms for labelled items. But after EP learning, the obtained Gaussian $Q(f_j)$ is defined not on labelled items, but on *all* the items (see Appendix). These Gaussian parameters will be used to update prior ϕ in the M-step as given in (3), and we update "pins" for each user separately via the same gradient method as in [3].

4 Empirical Study

We report some preliminary results on the EachMovie data set, which is popular for collaborative filtering research. For preprocessing we remove movies that are rated less than 50 times, and users who give less than 20 ratings. Then we end up with 809 movies, 4842 users and 334251 ratings. Since we don't have a feature representation for each movie, we select the 100 users with the most ratings as the true "users" in our model (i.e., regression functions), and treat the other users as features for each movie. A correlation kernel [1] is

[1] This is known as variational EM in the literature.

# Training Items	GPR	CGPR	GPOR	CGPOR
100	1.1971	1.1763	1.2366	1.2225
200	1.1283	1.1260	1.2126	1.1711
300	1.1042	1.1005	1.0907	1.0557
400	1.0970	1.0923	1.0454	1.0203

Table 1: Mean absolute errors for all the four models.

# Training Items	GPR	CGPR	GPOR	CGPOR
100	0.7432	0.7372	0.6239	0.6168
200	0.7224	0.7213	0.6087	0.5999
300	0.7155	0.7142	0.5862	0.5782
400	0.7125	0.7085	0.5760	0.5723

Table 2: Mean zero-one errors for all the four models.

defined for movies and used as the prior $\mathbf{\Sigma}$, in which we subtract the user mean and calculate cosine distance between two movie vectors. All the missing values for features are filled in as the corresponding user means. For the target values, about 60% of the entries are missing.

For learning the 100 functions (for the 100 users), we randomly select a subset of $\{100, 200, 300, 400\}$ movies for training, and prediction performance over the rest movies is averaged over all the users. Two comparison metrics are used: *mean absolute error* is the average deviation of the prediction from the target; *mean zero-one error* gives an error 1 to every incorrect prediction and then averages over all predictions. Prior function h is set to zero function initially. σ is initialized as 0.1 for GPR models and 1 for GPOR models. Table 1&2 give the performance for each approach. It can be seen that both the collaborative models outperform the corresponding individual models (in which we learn a GP model for each user separately). This means ranking can be improved if we model all the tasks jointly. If the prior kernel $\mathbf{\Sigma}$ is worse (e.g., calculated with less users), the differences will be larger. GPOR models are in general better than GPR models, except for mean absolute error with small training sets. This may be because of poor model fitting due to lack of data.

5 Discussion

This paper goes beyond ordinal regression for a single function and introduces a very general framework for modelling collaborative ordinal regression. A generative process of all the ordinal entries is presented, and learning can be done via a mixed EM and EP algorithm. Single-output ordinal regression is a special case of the proposed model.

The proposed EP algorithm allows any factorized form for the likelihood term $P(\mathbf{y}|\mathbf{X}, f, \theta)$. Recently Burges et al. proposed a neural network learner called RankNet [2], in which a logistic function is used to map pairwise outputs to probabilities. If we cast RankNet model into a Bayesian framework, it actually factorizes likelihood $P(\mathbf{y}|\mathbf{f}, \theta)$ with respect to pairs, i.e., $P(\mathbf{y}|\mathbf{f}, \theta) = \prod_{y_i \rhd y_k} P(y_i \rhd y_k | f(\mathbf{x}_i), f(\mathbf{x}_k), \theta) = \prod_{i,k} P_{ik}(o_{ik}, \theta)$.[2] Here $y_i \rhd y_k$ means "item i is preferred than item k", and $o_{ik} \equiv f(\mathbf{x}_i) - f(\mathbf{x}_k)$. P_{ik} is simply the logistic function $P_{ik}(o_{ik}, \theta) = \frac{\exp(o_{ik})}{1+\exp(o_{ik})}$. Note that we have an empty set for θ. This model is also ready to generalize to collaborative case, and the only difficulty for learning is that in E-step we need to calculate the first and second moments of the distri-

[2]This correspondence can be seen if we think $\max_\theta \mathcal{L}(\theta) = \min_\theta(-\log \mathcal{L}(\theta))$.

bution $z \sim \frac{\exp(z)}{1+\exp(z)}\mathcal{N}(z;\mu_z,\sigma_z^2)$. This involves one-dimensional integral and can be done via sampling method. One may argue that this is not a proper generative model since all the pairs should not be independent, but the experimental results in [2] somehow validate the model (with a neural network learner). An interesting future work is to investigate how this model behaves in a collaborative manner.

This paper focuses on a *transductive setting* of rank prediction, i.e., all the test items are known *a priori*. We just take their targets as missing data and come up with predicted means and variances after learning. For induction on previously unseen items, we need to map the learned kernel back using a similar approach as in [8]. It's interesting to empirically evaluate this approach.

Appendix: EP Learning

We describe EP learning for E-step. In the following we ignore the subindex j since we need to do EP for all the functions f_j's. To approximate the *a posteriori* distribution of f as a Gaussian $Q(f)$, in EP learning we take a factorized form for the likelihood term, i.e., $P(\mathbf{y}|\mathbf{X},f,\theta) = \prod_k t_k(f)$, and approximate each term $t_k(f)$ as a Gaussian sequentially [6]. Gaussian EP has been carefully investigated recently in [4], and it turns out that the critical terms for calculating EP approximation are $Z_k(\mathbf{u},\mathbf{C}) \equiv \int t_k(f)\mathcal{N}(f;\mathbf{u},\mathbf{C})\,df$ and its gradients $\mathbf{g}_k(\mathbf{u},\mathbf{C}) \equiv \partial \log Z_k(\mathbf{u},\mathbf{C})/\partial\mathbf{u}$ and $\mathbf{G}_k(\mathbf{u},\mathbf{C}) \equiv \partial \log Z_k(\mathbf{u},\mathbf{C})/\partial\mathbf{C}$. A careful manipulation shows that

$$\mathbf{g}_k(\mathbf{u},\mathbf{C}) = \mathbf{C}^{-1}(\langle f\rangle - \mathbf{u}), \mathbf{G}_k(\mathbf{u},\mathbf{C}) = \frac{1}{2}\mathbf{C}^{-1}\left(\langle ff^\top\rangle - \mathbf{u}\langle f\rangle^\top - \langle f\rangle\mathbf{u}^\top + \mathbf{u}\mathbf{u}^\top - \mathbf{C}\right)\mathbf{C}^{-1},$$

where $\langle\cdot\rangle$ denotes the expectation under distribution $f \sim t_k(f)\mathcal{N}(f;\mathbf{u},\mathbf{C})/Z_k(\mathbf{u},\mathbf{C})$. Calculation of these moments may be problematic for arbitrary term $t_k(f)$, but in this paper all these terms are one-dimensional, therefore we can at least use numerical integration method (e.g. Simpson quadrature) to calculate them. In some special cases, however, we can solve these integrals analytically. For instance in CGPR model, $t_k(f)$ are already Gaussian, so all the moments can be very easily calculated and the approximation is exact. For CGPOR model, the moments can be obtained through special properties of the error function (see [3] or [4]).

References

[1] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *ICML*, pages 65–72, 2004.

[2] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

[3] W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6:1019–1041, 2005.

[4] R. Herbrich. On Gaussian expectation propagation. 2005.

[5] P. McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society B*, 42(2):109–142, 1980.

[6] T. P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

[7] A. Schwaighofer, V. Tresp, and K. Yu. Hierarchical bayesian modelling with gaussian processes. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.

[8] K. Yu, V. Tresp, and A. Schwaighofer. Learning Gaussian processes from multiple tasks. In *ICML*, pages 1017–1024, 2005.