Sparse Preference Learning

Evgeni Tsivtsivadze and Tom Heskes
Institute for Computing and Information Sciences
Radboud University Nijmegen, The Netherlands
{evgeni,t.heskes}@science.ru.nl

Abstract

We propose a novel sparse preference learning/ranking algorithm. Our algorithm approximates the true utility function by a weighted sum of basis functions using the squared loss on pairs of data points, and is a generalization of the matching pursuit method. It can operate both in a supervised and a semi-supervised setting and allows efficient search for multiple, near-optimal solutions. In our experiments we demonstrate that the proposed algorithm outperforms several state-of-the-art learning methods when taking into account unlabeled data and performs comparably in a supervised learning scenario, while providing sparser solution.

1 Introduction

Learning preference relations involves prediction of ordering of the data points rather than prediction of a single numerical value as in the case of regression or a class label as in the case of a classification task. The ranking problem can be considered as a special case of preference learning when a strict order is defined over all data points. Despite notable progress in the development and application of preference learning/ranking algorithms (e.g. [5]), so far the emphasis was mainly on improving the learning performance of the method (e.g. [2, 1]) and much less is known about the models that focus in addition on interpretability and sparseness of the ranking solution. Besides interpretability, sparse models also lead to notably faster prediction times (that is an absolute necessity for a wide range of applications such as e.g. search engines), compared to the non-sparse counterparts. A ranking method that can lead to sparse solutions is RankSVM [6]. However, in RankSVM sparsity control is not explicit and the produced models are usually far from being interpretable. Also note, that frequently ranking algorithms are not directly applicable to more general preference learning task or can become computationally expensive.

In this paper we propose a sparse preference learning/ranking algorithm. Our method is a generalization of the (kernel) matching pursuit algorithm [9] and it approximates true utility function by a weighted sum of basis functions using squared loss on pairs of data points. Unlike existing methods our algorithm allows explicit control over sparsity of the model and can be applied to ranking and preference learning problems. Furthermore, an extension of the algorithm allows us to efficiently search for several near-optimal solutions instead of a single one. We show that our algorithm can operate in supervised or semi-supervised setting, leads to sparse solutions, and improved performance compared to several baseline methods.

2 Problem Setting

Let $\mathcal{X}$ be a set of instances and $\mathcal{Y}$ be a set of labels. We consider the *label ranking* task [5, 3] namely, we want to predict for any instance $\mathbf{x} \in \mathcal{X}$ a preference relation $\mathcal{P}_{\mathbf{x}} \subseteq \mathcal{Y} \times \mathcal{Y}$ among the set of labels $\mathcal{Y}$. We assume that the true preference relation $\mathcal{P}_{\mathbf{x}}$ is transitive and asymmetric for each instance $\mathbf{x} \in \mathcal{X}$. Our training set $\{(\mathbf{q}_i, s_i)\}_{i=1}^{n}$ contains the data points $(\mathbf{q}_i, s_i) = ((\mathbf{x}_i, y_i), s_i) \in (\mathcal{X} \times \mathcal{Y}) \times \mathbb{R}$

that are an instance-label tuple $\mathbf{q}_i = (\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ and its score $s_i \in \mathbb{R}$. We define the pair of data points $((\mathbf{x}, y), s)$ and $((\mathbf{x}', y'), s')$ to be *relevant*, iff $\mathbf{x} = \mathbf{x}'$ and *irrelevant* otherwise.

As an example, consider an information retrieval task where every query is associated with the set of retrieved documents. The intersection of the retrieved documents associated with different queries can be either empty or non-empty. We are usually interested in ranking the documents that are associated with a single query (the one that has retrieved the documents). Thus, ranks between documents retrieved by different queries are *not relevant* for this task, whereas those documents retrieved by the same query are *relevant*.

Given a relevant pair $((\mathbf{x}, y), s)$ and $((\mathbf{x}, y'), s')$, we say that instance $\mathbf{x}$ *prefers* label y to y', iff $s > s'$. If $s = s'$, the labels are called *tied*. Accordingly, we write $y \succ_{\mathbf{x}} y'$ if $s > s'$ and $y \sim_{\mathbf{x}} y'$ if $s = s'$. Finally, we define our training set $\mathcal{T} = (Q, \mathbf{s}, W)$, where $Q = (\mathbf{q}_1, \ldots, \mathbf{q}_n)^t \in (\mathcal{X} \times \mathcal{Y})^n$ is the vector of instance-label training tuples and $\mathbf{s} = (s_1, \ldots, s_n)^t \in \mathbb{R}^n$ is the corresponding vector of scores. The W matrix defines a preference graph and incorporates information about relevance of a particular data point to the task, e.g. $[W]_{i,j} = 1$, if $(\mathbf{q}_i, \mathbf{q}_j), 1 \leq i, j \leq n, i \neq j$, are relevant and is 0 otherwise.

Informally, the goal of our ranking task is to find a *label ranking function* such that the ranking $\mathcal{P}_{f,\mathbf{x}} \subseteq \mathcal{Y} \times \mathcal{Y}$ induced by the function for any instance $\mathbf{x} \in \mathcal{X}$ is a good "prediction" of the true preference relation $\mathcal{P}_{\mathbf{x}} \subseteq \mathcal{Y} \times \mathcal{Y}$. Formally, we search for the function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ mapping each instance-label tuple $(\mathbf{x}, y)$ to a real value representing the (predicted) relevance of the label y with respect to the instance $\mathbf{x}$. To measure how well a hypothesis f is able to predict the preference relations $\mathcal{P}_{\mathbf{x}}$ for all instances $\mathbf{x} \in \mathcal{X}$, we consider the following cost function (disagreement error) that captures the amount of incorrectly predicted pairs of relevant training data points: $d(f, \mathcal{T}) = \frac{1}{2} \sum_{i,j=1}^{n} [W]_{i,j} \left| \operatorname{sign}(s_i - s_j) - \operatorname{sign}(f(\mathbf{q}_i) - f(\mathbf{q}_j)) \right|$, where $\operatorname{sign}(\cdot)$ denotes the signum function.

3 Ranking Pursuit

In this section we tailor the kernel matching pursuit algorithm [9] to the specific setting of preference learning/ranking problem. Considering the training set $\mathcal{T} = (Q, \mathbf{s}, W)$ and a dictionary of functions $\mathcal{D} = \{k_1, \ldots, k_N\}$, where N is number of functions in the dictionary, we are interested in finding sparse approximation of the prediction function $f_P(\mathbf{q}) = \sum_{p=1}^{P} a_p k_{\gamma_p}(\mathbf{q})$ using the basis functions $\{k_1, \ldots, k_P\} \subset \mathcal{D}$ and the coefficients $\{a_1, \ldots, a_P\} \in \mathbb{R}^P$. The order of the dictionary functions as they appear in the expansion is given by a set of indices $\{\gamma_1, \ldots, \gamma_P\}$, where $\gamma \in \{1, \ldots, N\}$. We note that basis functions in our case are the kernel functions, similar to [9]. We will use notation $\mathbf{f}_P = (f_P(\mathbf{q}_1), \ldots, f_P(\mathbf{q}_n))$ to represent the n-dimensional vector that corresponds to the evaluation of the function on the training points. We also define $\mathbf{r} = \mathbf{s} - \mathbf{f}_P$ to be the residue. The basis functions and the corresponding coefficients are to be chosen such that they minimize an approximation of the disagreement error: $c(f_P, \mathcal{T}) = \frac{1}{2} \sum_{i,j=1}^{n} [W]_{i,j} \Big((s_i - s_j) - (f_P(\mathbf{q}_i) - f_P(\mathbf{q}_j))\Big)^2$, which in matrix form can be written as $c(\mathbf{f}_P, \mathcal{T}) = (\mathbf{s} - \mathbf{f}_P)^t L (\mathbf{s} - \mathbf{f}_P)$, where L is the Laplacian matrix of the graph W.

The ranking pursuit starts at stage 0, with $\mathbf{f}_0$, and recursively appends functions to an initially empty basis, at each stage of training to reduce the approximation of the ranking error. Given $\mathbf{f}_p$ we build $\mathbf{f}_{p+1}(a, \gamma) = \mathbf{f}_p + a\mathbf{k}_\gamma$, by searching for $\gamma \in \{1, \ldots, N\}$ and $a \in \mathbb{R}$ such that at every step (the residue of) the error is minimized:

$$J(a, \gamma) = c(\mathbf{f}_{p+1}(a, \gamma), \mathcal{T}) = (\mathbf{s} - \mathbf{f}_{p+1}(a, \gamma))^t L (\mathbf{s} - \mathbf{f}_{p+1}(a, \gamma)) = (\mathbf{r}_p - a\mathbf{k}_\gamma)^t L (\mathbf{r}_p - a\mathbf{k}_\gamma),$$

where we use a notation $k_{\gamma i} = k(\mathbf{q}_\gamma, \mathbf{q}_i)$ and $\mathbf{k}_\gamma = (k_{\gamma 1}, \ldots, k_{\gamma n})^t$. The a that minimizes $J(a, \gamma)$ for a given γ reads $a = (\mathbf{k}_\gamma^t L \mathbf{k}_\gamma)^{-1} \mathbf{k}_\gamma^t L \mathbf{r}_p$. The set of basis functions and coefficients obtained at every iteration of the algorithm is suboptimal. This can be corrected by back-fitting procedure using a least-squares approximation of the disagreement error. The optimal value of parameter N, that can be considered as a "regularization" parameter of the algorithm, is estimated using a cross-validation procedure.

Learning Multiple Near-Optimal Solutions

In this subsection we formulate extension of the ranking pursuit algorithm that can efficiently use unscored data to improve performance of the algorithm. The main idea behind our approach is to construct multiple, near-optimal, "sparse" ranking functions that give a small error on the scored data and whose predictions agree on the unscored part.

Let us consider M different feature spaces $\mathcal{H}_1, \ldots, \mathcal{H}_M$ that can be constructed from different data point descriptions (i.e., different features) or by using different kernel functions. Similar to [9] we consider $\mathcal{H}$ to be a RKHS. In addition to the training set $\mathcal{T} = (Q, \mathbf{s}, W)$ originating from a set $\{(\mathbf{q}_i, s_i)\}_{i=1}^n$ of data points *with* scoring information, we also have a training set $\overline{\mathcal{T}} = (\overline{Q}, \overline{W})$ from a set $\{\overline{\mathbf{q}}_i\}_{i=1}^l$ of data points *without* scoring information, $\overline{Q} = (\overline{\mathbf{q}}_1, \ldots, \overline{\mathbf{q}}_l)^t \in (\mathcal{X} \times \mathcal{Y})^l$, and the corresponding adjacency matrix $\overline{W}$. To avoid misunderstandings with the definition of the label ranking task, we will use the terms "scored" instead of "labeled" and "unscored" instead of "unlabeled". We search for the functions $F_P = (f_P^{(1)}, \ldots, f_P^{(M)}) \in \mathcal{H}_1 \times \ldots \times \mathcal{H}_M$, minimizing

$$\widetilde{c}(F_P, \mathcal{T}, \overline{\mathcal{T}}) = \sum_{v=1}^{M} c(f_P^{(v)}, \mathcal{T}) + \nu \sum_{v,u=1}^{M} \overline{c}(f_P^{(v)}, f_P^{(u)}, \overline{\mathcal{T}}), \tag{1}$$

where $\nu \in \mathbb{R}^+$ is a regularization parameter and where $\overline{c}$ is the loss function measuring the disagreement between the prediction functions of the views on the unscored data:

$$\overline{c}(f_P^{(v)}, f_P^{(u)}, \overline{\mathcal{T}}) \quad = \quad \frac{1}{2} \sum_{i,j=1}^{l} [\overline{W}]_{i,j} \Big(\big(f_P^{(v)}(\overline{\mathbf{q}}_i) - f_P^{(v)}(\overline{\mathbf{q}}_j)\big) - \big(f_P^{(u)}(\overline{\mathbf{q}}_i) - f_P^{(u)}(\overline{\mathbf{q}}_j)\big) \Big)^2.$$

Although we have used unscored data in our formulation, we note that the algorithm can also operate in a purely supervised setting: It will not only minimize the error on the scored data but also enforce agreement among the prediction functions constructed from different views. The prediction functions $f_P^{(v)} \in \mathcal{H}_v$ of (1) for $v = 1, \ldots, M$ have the form $f_P^{(v)}(\mathbf{q}) = \sum_{p=1}^{P} a_p^{(v)} k_{\gamma_v p}^{(v)}(\mathbf{q})$ with corresponding coefficients $\{a_1^{(v)}, \ldots, a_P^{(v)}\} \in \mathbb{R}^P$. Let $\bar{L}$ denote the Laplacian matrix of the graph $\overline{W}$. Using a similar approach as in Sec. 3 we can write the objective function as

$$\begin{aligned} J(\mathbf{a}, \boldsymbol{\gamma}) \quad &= \quad \widetilde{c}(F_{p+1}(\mathbf{a}, \boldsymbol{\gamma}), \mathcal{T}, \overline{\mathcal{T}}) = \sum_{v=1}^{M} (\mathbf{r}_p - a^{(v)} \mathbf{k}_{\gamma_v}^{(v)})^t L (\mathbf{r}_p - a^{(v)} \mathbf{k}_{\gamma_v}^{(v)}) \\ &+ \quad \nu \sum_{v,u=1}^{M} (a^{(v)} \bar{\mathbf{k}}_{\gamma_v}^{(v)} - a^{(u)} \bar{\mathbf{k}}_{\gamma_u}^{(u)})^t \bar{L} (a^{(v)} \bar{\mathbf{k}}_{\gamma_v}^{(v)} - a^{(u)} \bar{\mathbf{k}}_{\gamma_u}^{(u)}), \end{aligned}$$

where $\mathbf{a} = (a^{(1)}, \ldots, a^{(M)})^t \in \mathbb{R}^M$, $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_M)$ with $\gamma_v \in \{1, \ldots, N\}$, and $\bar{\mathbf{k}}_\gamma$ is the basis vector expansion on unscored data with $\bar{k}_{\gamma i} = k(\overline{\mathbf{q}}_\gamma, \overline{\mathbf{q}}_i)$. By taking partial derivatives with respect to the coefficients in each view (for clarity we denote $\mathbf{k}_{\gamma_v}^{(v)}$ and $\bar{\mathbf{k}}_{\gamma_v}^{(v)}$ as $\mathbf{k}^{(v)}$and $\bar{\mathbf{k}}^{(v)}$, respectively) and defining $g_\nu^{(v)} = 2\nu(M-1)\bar{\mathbf{k}}^{(v)t} \bar{L} \bar{\mathbf{k}}^{(v)}$ and $g^{(v)} = \mathbf{k}^{(v)t} L \mathbf{k}^{(v)}$, we obtain

$$\frac{d}{da^{(v)}} J(\mathbf{a}, \boldsymbol{\gamma}) = 2(g^{(v)} + g_\nu^{(v)}) a^{(v)} - 2\mathbf{k}^{(v)t} L \mathbf{r}_p - 4\nu \sum_{u=1, u \neq v}^{M} \bar{\mathbf{k}}^{(v)t} \bar{L} \bar{\mathbf{k}}^{(u)} a^{(u)}.$$

At the optimum we have $\frac{d}{da^{(v)}} J(\mathbf{a}, \boldsymbol{\gamma}) = 0$ for all views, thus, we get the exact solution by solving

$$\begin{pmatrix} g^{(1)} + g_\nu^{(1)} & -2\nu \bar{\mathbf{k}}^{(1)t} \bar{L} \bar{\mathbf{k}}^{(2)} & \ldots \\ -2\nu \bar{\mathbf{k}}^{(2)} \bar{L} \bar{\mathbf{k}}^{(1)} & g^{(2)} + g_\nu^{(2)} & \ldots \\ \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} a^{(1)} \\ a^{(2)} \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{k}^{(1)t} L \mathbf{r}_p \\ \mathbf{k}^{(2)t} L \mathbf{r}_p \\ \vdots \end{pmatrix}$$

with respect to the coefficients in each view. Note that the left-hand side matrix is positive definite by construction and, therefore, invertible. Once the coefficients are estimated, multiple solutions can

Require: Training set with scored and unscored data - $\mathcal{T}$, $\overline{\mathcal{T}}$, dictionary of functions - $\mathcal{D}$, number of the basis functions - P, and the co-regularization parameter - ν.

Ensure: Construct residue vector $\mathbf{r}$

1: **for** $p = 1, \ldots, P$ (or until performance on the validation set stops improving) **do**

2: $\boldsymbol{\gamma}_p = \operatorname{argmin}_{\boldsymbol{\gamma}} J(\mathbf{a}^*(\boldsymbol{\gamma}), \boldsymbol{\gamma})$

3: Compute $\mathbf{a}^*(\boldsymbol{\gamma}_p) = (B + C)^{-1}\mathbf{e}$ using the matrices (notation from Sec. 3):

$$B = \begin{pmatrix} g^{(1)} & 0 & \ldots \\ 0 & g^{(2)} & \ldots \\ \vdots & \vdots & \ddots \end{pmatrix} \quad \mathbf{e} = \begin{pmatrix} \mathbf{k}^{(1)t}_{\gamma_1} L \mathbf{r}_p \\ \mathbf{k}^{(2)t}_{\gamma_2} L \mathbf{r}_p \\ \vdots \end{pmatrix} \quad C = \begin{pmatrix} g^{(1)}_{\nu} & -2\nu \bar{\mathbf{k}}^{(1)t}_{\gamma_1} \bar{L} \bar{\mathbf{k}}^{(2)}_{\gamma_2} & \ldots \\ -2\nu \bar{\mathbf{k}}^{(2)t}_{\gamma_2} \bar{L} \bar{\mathbf{k}}^{(1)}_{\gamma_1} & g^{(2)}_{\nu} & \ldots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

4: Set $\mathbf{a} = \mathbf{a}^*(\boldsymbol{\gamma}_p)$ and compute new residual $\mathbf{r}_{p+1} = \mathbf{r}_p - \frac{1}{M} \sum_{v=1}^{M} a^{(v)} \mathbf{k}^{(v)}_{\gamma_v}$

5: **end for**

6: Compute prediction:

$$f_P(\mathbf{q}) = \frac{1}{M} \sum_{v=1}^{M} \sum_{p=1}^{P} a^{(v)}_p k^{(v)}_{\gamma_v p}(\mathbf{q})$$

Figure 1: Semi-supervised ranking pursuit algorithm.

be obtained using the prediction functions constructed for each view. We can also consider a single prediction function that is given, for example, by the average of the functions for all views. The overall complexity of the standard ranking pursuit algorithm is $\mathcal{O}(Pn^2)$, thus, there is no increase in computational time compared to the kernel matching pursuit algorithm in the supervised setting [9]. The semi-supervised version of the ranking pursuit algorithm requires $\mathcal{O}(Pn^M(M^3 + M^2 l))$ time, which is linear in the number of unscored data points[1]. The pseudo-code for the algorithm is presented in Figure 1.

4 Experiments

We perform a set of experiments on the publicly available Jester joke dataset[2]. The task we address is the prediction of the joke preferences of a user based on the preferences of other users. The dataset contains 4.1M ratings in the range from -10.0 to $+10.0$ of 100 jokes assigned by a group of 73421 users. Our experimental setup is similar to that of [2]. We have grouped the users into three groups according to the number of jokes they have rated: $20 - 40$ jokes, $40 - 60$ jokes, and $60 - 80$ jokes. The test users are randomly selected among the users who had rated between 50 and 300 jokes. For each test user half of the preferences is reserved for training and half for testing. The preferences are derived from the differences of the ratings the test user gives to jokes, e.g. a joke with higher score is preferred over the joke with lower score. The features for each test user are generated as follows: A set of 300 reference users is selected at random from one of the three groups and their ratings for the corresponding jokes are used as a feature values. In case user has not rated the joke the median of his/her ratings is used as the feature value. The experiment is done for 300 different test users and the average performance is recorded. Finally, we repeat complete experiment ten times with a different set of 300 test users selected at random. We report the average value over the ten runs for each of the three groups.

In this experiment we compare performance of the ranking pursuit algorithm to several algorithms, namely kernel matching pursuit [9], RankSVM [6], RLS [8], and RankRLS [7] in terms of the disagreement error. In all algorithms we use a Gaussian kernel where the width parameter is chosen from the set $\{2^{-15}, 2^{-14} \ldots, 2^{14}, 2^{15}\}$ and other parameters (e.g. stopping criteria) are chosen by

[1] In semi-supervised learning usually $n \ll l$, thus, linear complexity in the number of unscored data points is beneficial. We note that complexity of the algorithm can be further reduced to $\mathcal{O}(PM^3 nl)$ by forcing the indices of the nonzero coefficients in the different views to be the same.

[2] Available at `http://www.ieor.berkeley.edu/~goldberg/jester-data/`.

Table 1: Performance comparison of the kernel matching pursuit, RLS, RankSVM, RankRLS, and ranking pursuit algorithms in the supervised learning experiment conducted on Jester joke dataset. Normalized version of the disagreement error is used as a performance evaluation measure.

METHOD	$\mathbf{20-40}$	$\mathbf{40-60}$	$\mathbf{60-80}$
RLS	0.425	0.419	0.383
MATCHING PURSUIT	0.428	0.417	0.381
RANKSVM	0.412	0.404	0.372
RANKRLS	0.409	0.407	0.374
RANKING PURSUIT	0.410	0.404	0.373

Table 2: Performance comparison of the kernel matching pursuit, RLS, RankSVM, RankRLS, ranking pursuit, and semi-supervised ranking pursuit algorithms in the semi-supervised learning experiment conducted on Jester joke dataset. Supervised learning methods are trained only on the scored part of the dataset. Normalized version of the disagreement error is used as a performance evaluation measure.

METHOD	$\mathbf{20-40}$	$\mathbf{40-60}$	$\mathbf{60-80}$
RLS	0.449	0.434	0.405
MATCHING PURSUIT	0.451	0.433	0.404
RANKSVM	0.428	0.417	0.391
RANKRLS	0.429	0.418	0.393
RANKING PURSUIT	0.428	0.417	0.393
SS RANKING PURSUIT	0.419	0.411	0.381

taking the average over the performances on a hold out-set. The hold-out set is created similarly as the corresponding training/test set.

The results of the collaborative filtering experiment are included in Table 1. It can be observed that ranking based approaches in general outperform regression methods. Although performance of the ranking pursuit algorithm is similar to that of the RankSVM and RankRLS algorithms, obtained solutions are on average 30% sparser.

To evaluate performance of the semi-supervised extension of the ranking pursuit algorithm we construct datasets similarly as in the supervised learning experiment with the following modification: To simulate unscored data, for each test user we leave only a half of his/her preferences from the training set to be available for learning. Using this training set we construct two views, each containing a half of the scored and a half of the unscored data points. The rest of experimental setup follows previously described supervised learning setting. The results of this experiment are included in Table 2. We observe notable improvement in performance of the semi-supervised ranking pursuit algorithm compared to all baseline methods. This improvement is statistically significant according to Wilcoxon signed-rank test [4] with 0.05 as a significance threshold.

5 Conclusions

We propose a sparse preference learning/ranking algorithm and its semi-supervised extension. Our algorithm allows explicit control over sparsity and is naturally applicable in situations when one is interested in obtaining several near-optimal solutions. The experiments demonstrate that in the supervised setting our algorithm outperforms regression methods such as kernel matching pursuit, RLS and performs comparably to the RankRLS and RankSVM algorithms, while having sparser solution. In semi-supervised setting the proposed algorithm notably outperforms all baseline methods. In the future we aim to apply our algorithm in other domains and will examine different aggregation techniques for multiple sparse solutions.

Acknowledgments

We acknowledge support from the Netherlands Organization for Scientific Research (NWO), in particular `Learning2Reason` and `Vici` grants (639.023.604).

References

[1] Adriana Birlutiu, Perry Groot, and Tom Heskes. Multi-task preference learning with an application to hearing aid personalization. *Neurocomputing*, 73(7-9):1177–1185, 2010.

[2] Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. Magnitude-preserving ranking algorithms. In Zoubin Ghahramani, editor, *Proceedings of the 24th Annual International Conference on Machine Learning*, pages 169–176, New York, NY, USA, 2007. ACM.

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

[4] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[5] Johannes Fürnkranz and Eyke Hüllermeier (Eds.). *Preference Learning*. Springer, Cambridge, Massachusetts, 2010.

[6] Thorsten Joachims. A support vector method for multivariate performance measures. In *22nd International Conference on Machine Learning*, pages 377–384, New York, NY, USA, 2005. ACM.

[7] Tapio Pahikkala, Evgeni Tsivtsivadze, Antti Airola, Jouni Järvinen, and Jorma Boberg. An efficient algorithm for learning to rank from preference graphs. *Machine Learning*, 75(1):129–165, 2009.

[8] Ryan Rifkin, Gene Yeo, and Tomaso Poggio. Regularized least-squares classification. In J.A.K. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, editors, *Advances in Learning Theory: Methods, Model and Applications*, pages 131–154, Amsterdam, 2003. IOS Press.

[9] Pascal Vincent and Yoshua Bengio. Kernel matching pursuit. *Machine Learning*, 48(1-3):165–187, 2002.