Kernel Principal Component Ranking: Robust Ranking on Noisy Data

Evgeni Tsivtsivadze, Botond Cseke, and Tom Heskes

Institute for Computing and Information Sciences, Radboud University Nijmegen,
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands
firstname.lastname@science.ru.nl

Abstract. We propose the kernel principal component ranking algorithm (KPCRank) for learning preference relations. The algorithm can be considered as an extension of nonlinear principal component regression applicable to preference learning task. It is particularly suitable for learning from noisy datasets where a lower dimensional data representation preserves most expressive features. In many cases near-linear dependence of regressors (multicollinearity) can notably decrease performance of the learning algorithm, however, KPCRank can effectively deal with this situation. It is accomplished by projecting the data onto p-principal components in the feature space defined by a positive definite kernel and consecutive learning of the ranking function. Despite the fact that the number of the pairwise preferences is quadratic, the training time of KPCRank scales linearly with the number of data points in the training set and is equal to that of the principal component regression. We compare the algorithm to several ranking and regression methods, including probabilistic regression on pairwise comparison data. Our experiments demonstrate that the performance of KPCRank is better than that of the baseline methods, when learning to rank from the data corrupted by noise.

1 Introduction

The task of learning preference relations (see e.g. [5]) has received significant attention in machine learning literature. This paper primarily concerns the task of ranking, which is a special case of a preference learning task when a total order is associated with the set of data points under consideration. Both the preference learning and the ranking tasks can be formulated as the problems where the aim is to learn a function capable of arranging data points according to a given preference relation. When comparing two data points, the function is able to evaluate whether the first point is preferred over the second one. To learn this function we propose the kernel principal component ranking (KPCRank) algorithm.

The algorithm can be considered as an extension of the nonlinear principal component regression applicable to the preference learning task. Similar to the kernel principal component regression (KPCR) [14], KPCRank is particularly

suitable for learning from noisy datasets where lower dimensional data representation preserves most expressive features. By projecting the original data onto the components with higher eigenvalues we can discard the noise contained in the original data. We derive the algorithm and describe an efficient way for selecting optimal number of the principal components suitable for the task in question. To demonstrate that the algorithm can learn to rank well from noisy data, we evaluate KPCRank on the dataset corrupted by noise.

We consider a label ranking problem setting in which we are given a training data consisting of the scored data points, that is, each input data point is assigned a real valued score indicating its goodness. The pairwise preferences between these data points are then determined by the differences of the scores. Learning these preferences can also be treated as an extension of pairwise classification [7] where class label indicates direction of the preference. For example, in [2, 1] the pairwise preference approach is used together with Gaussian processes to learn preference relations. Pairwise approach is also used with support vector machines (SVM) for learning to rank. For example, the RankSVM algorithm was proposed in [8] to rerank the results obtained from a search engine.

We evaluate our algorithm on a parse ranking task that is a common problem in natural language processing (see e.g. [3]). In this task, the aim is to rank a set of parses associated with a single sentence, based on some goodness criteria. We consider the parse ranking task as label ranking. However, in the parse ranking task the labels (i.e. the parses of a sentence) are instance-specific. That is, for each sentence, we have a different set of labels, while in the conventional label ranking setting labels are not instance specific [5]. As baseline methods in the conducted experiments we consider the regularized least-squares (RLS) [13], the RankRLS [11], and the kernel principal component regression (KPCR) [14] algorithms. Furthemore, we compare the KPCRank algorithm to the probabilistic regression on pairwise preference data that is generated using sinc function. The results show that the performance of the KPCRank algorithm is better than that of the baseline methods, when learning to rank from the data corrupted by noise.

2 Label Ranking

Let $\mathcal{X}$ be a set of instances and $\mathcal{Y}$ be a set of labels. In the label ranking setting [4, 5], we would like to predict for any instance $\boldsymbol{x} \in \mathcal{X}$ a preference relation $\mathcal{P}_{\boldsymbol{x}} \subseteq \mathcal{Y} \times \mathcal{Y}$ among the set of labels $\mathcal{Y}$. An element $(y, y') \in \mathcal{P}_{\boldsymbol{x}}$ means that the instance $\boldsymbol{x}$ prefers the label y compared to y', also written as $y \succ_{\boldsymbol{x}} y'$. We assume that the preference relation $\mathcal{P}_{\boldsymbol{x}}$ is transitive and asymmetric for each instance $x \in \mathcal{X}$. As a training information, we are given a finite set $\{(\boldsymbol{q}_i, s_i)\}_{i=1}^{n}$ of n data points, where each data point $(\boldsymbol{q}_i, s_i) = ((\boldsymbol{x}_i, y_i), s_i) \in \mathcal{Q} \times \mathbb{R}$ consists of an instance-label tuple $\boldsymbol{q}_i = (\boldsymbol{x}_i, y_i) \in \mathcal{Q}$, where $\mathcal{Q} = \mathcal{X} \times \mathcal{Y}$ and the score $s_i \in \mathbb{R}$. We consider two data points $(\boldsymbol{q}, s)$ and $(\boldsymbol{q}', s')$ to be relevant, iff $\boldsymbol{x} = \boldsymbol{x}'$. For relevant data points, instance $\boldsymbol{x}$ prefers label y to y', if $s > s'$. If $s = s'$, the labels are called tied. Accordingly, we write $y \succ_{\boldsymbol{x}} y'$ if $s > s'$ and $y \sim_{\boldsymbol{x}} y'$ if $s = s'$. To be able to

incorporate the relevance information we define an undirected preference graph which is determined by its adjacency matrix W such that $W_{ij} = 1$ if $(\boldsymbol{q}_i, \boldsymbol{q}_j)$ are relevant and $W_{ij} = 0$ otherwise. Furthermore, let $Q = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n)^t \in \mathcal{Q}^n$ be the vector of instance-label training tuples and $\boldsymbol{s} = (s_1, \ldots, s_n)^t \in \mathbb{R}^n$ the corresponding vector of scores. Given these definitions, our training set is the tuple $\mathcal{T} = (Q, W, \boldsymbol{s})$. We also note that our definitions allow considering bipartite ranking, ordinal regression, and label ranking setting at the same time.

We define mapping from the input space $\mathcal{Q}$ to some (higher dimensional) feature space $\mathcal{F}$. In many cases, the number of training data points is much smaller than the number of dimensions m of the feature space and we can write $\mathcal{F} = \mathbb{R}^m$, where $m \gg n$. Further, let

$$\Phi : \mathcal{Q} \to \mathcal{F}.$$

The inner product

$$k(\boldsymbol{q}, \boldsymbol{q}') = \langle \Phi(\boldsymbol{q}), \Phi(\boldsymbol{q}') \rangle$$

of the mapped instance-label pairs is called a kernel function. We also denote the sequence of feature mapped inputs as

$$\Phi(Q) = (\Phi(\boldsymbol{q}_1), \ldots, \Phi(\boldsymbol{q}_n)) \in (\mathcal{F}^n)^t$$

for all $Q \in (\mathcal{Q}^n)^t$. We define the symmetric kernel matrix $K \in \mathbb{R}^{n \times n}$, where $\mathbb{R}^{n \times n}$ denotes the set of real matrices of dimension $n \times n$, as

$$K = \Phi(Q)^t \Phi(Q) = \begin{pmatrix} k(\boldsymbol{q}_1, \boldsymbol{q}_1) & \cdots & k(\boldsymbol{q}_1, \boldsymbol{q}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{q}_n, \boldsymbol{q}_1) & \cdots & k(\boldsymbol{q}_n, \boldsymbol{q}_n) \end{pmatrix}.$$

Unless stated otherwise, we assume that the kernel matrix is strictly positive definite.

3 Dimensionality Reduction

Before presenting the KPCRank algorithm we briefly describe the procedure for kernel principal component analysis (KPCA), following [15]. KPCA allows us to perform standard PCA in the higher dimensional space $\mathcal{F}$ using the kernel function described in Section 2. After the initial mapping, the data falls onto some hyperplane in $\mathcal{F}$ and the extracted principal components will map onto manifolds in the lower dimensional space. The KPCA algorithm boils down to diagonalization of the covariance matrix

$$C = \frac{1}{m} \sum_{i=1}^{n} \Phi(\boldsymbol{q}_i) \Phi(\boldsymbol{q}_i)^t = \frac{1}{m} \Phi(Q) \Phi(Q)^t,$$

where the $\Phi(\boldsymbol{q}_i)$ are the centered mappings of the individual instance-label pairs. To find the first principal component we need to solve the eigenvalue equation

$$C\boldsymbol{v} = \lambda \boldsymbol{v}. \tag{1}$$

The key observation made in [15] states that all solutions $\boldsymbol{v}$ with $\lambda \neq 0$ can be written as a linear combination of $\Phi(\boldsymbol{q}_i)$. Therefore, there are some $a_i \in \mathbb{R}$ such that $\boldsymbol{v} = \sum_{i=1}^{n} a_i \Phi(\boldsymbol{q}_i)$. Then Equation (1) can be written as

$$\begin{aligned}
&\frac{1}{m}\Phi(Q)\Phi(Q)^t\boldsymbol{v} = \lambda\boldsymbol{v} \\
&\frac{1}{m}\Phi(Q)\Phi(Q)^t\Phi(Q)\boldsymbol{a} = \lambda\Phi(Q)\boldsymbol{a} \\
&\frac{1}{m}\Phi(Q)^t\Phi(Q)\Phi(Q)^t\Phi(Q)\boldsymbol{a} = \lambda\Phi(Q)^t\Phi(Q)\boldsymbol{a} \\
&\frac{1}{m}K\boldsymbol{a} = \lambda\boldsymbol{a},
\end{aligned}$$

where $\boldsymbol{a} \in \mathbb{R}^n$. Thus, we have arrived to the equivalent eigenvalue problem that requires diagonalization of K instead of C. Now we can compute the projection of the mapped instance-label pair $\Phi(\boldsymbol{q})$ onto $l-th$ eigenvector $\boldsymbol{v}^l$

$$\langle \boldsymbol{v}^l, \Phi(\boldsymbol{q})\rangle = \frac{1}{\sqrt{m\lambda_l}} \sum_{i=1}^{n} \boldsymbol{a}_i^l \langle \Phi(\boldsymbol{q}_i)\Phi(\boldsymbol{q})\rangle = \frac{1}{\sqrt{m\lambda_l}} \sum_{i=1}^{n} \boldsymbol{a}_i^l k(\boldsymbol{q}_i, \boldsymbol{q}). \tag{2}$$

If we denote by $V \in \mathbb{R}^{m\times p}$ the matrix consisting of the columns of the eigenvectors $\{\boldsymbol{v}^i\}_{i=1}^{p}$ of the covariance matrix C, by $\bar{\Lambda} \in \mathbb{R}^{p\times p}$ the diagonal matrix of the eigenvalues corresponding to the extracted eigenvectors of K, and by $\bar{V} \in \mathbb{R}^{n\times p}$ the matrix of extracted eigenvectors $\{\boldsymbol{a}^i\}_{i=1}^{p}$ of the kernel matrix K, then the projection of the instance-label pairs Q onto first p eigenvectors can be written as

$$\Phi(Q)^t V = \Phi(Q)^t\Phi(Q)\bar{V}\bar{\Lambda}^{-\frac{1}{2}} = K\bar{V}\bar{\Lambda}^{-\frac{1}{2}}. \tag{3}$$

4 Kernel Principal Component Ranking Algorithm

A ranking function is a function $f : \mathcal{Q} \to \mathbb{R}$ mapping the instance-label pair $\boldsymbol{q}$ to a real value representing the relevance of the label y with respect to the instance $\boldsymbol{x}$. This induces for any instance $\boldsymbol{x} \in \mathcal{X}$ a transitive preference relation $\mathcal{P}_{f,\boldsymbol{x}} \subseteq \mathcal{Y} \times \mathcal{Y}$ with $(y, y') \in \mathcal{P}_{f,\boldsymbol{x}} \Leftrightarrow f(\boldsymbol{q}) > f(\boldsymbol{q}')$. Informally, the goal of our ranking task is to find a label ranking function $f : \mathcal{Q} \to \mathbb{R}$ such that the ranking $\mathcal{P}_{f,\boldsymbol{x}} \subseteq \mathcal{Y}\times\mathcal{Y}$ induced by the function for any instance $\boldsymbol{x} \in \mathcal{X}$ is a good prediction for the true preference relation $\mathcal{P}_{\boldsymbol{x}} \subseteq \mathcal{Y} \times \mathcal{Y}$.

Let us define $\mathbb{R}^{\mathcal{Q}} = \{f : \mathcal{Q} \to \mathbb{R}\}$ and let $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{Q}}$ be the hypothesis space of possible ranking functions. To measure how well a hypothesis $f \in \mathcal{H}$ is able to predict the preference relations $\mathcal{P}_{\boldsymbol{x}}$ for all instances $\boldsymbol{x} \in \mathcal{X}$, we consider the following cost function that captures the amount of incorrectly predicted pairs of the relevant training data points:

$$d(f, \mathcal{T}) = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij} \left| sign\big(s_i - s_j\big) - sign\big(f(\boldsymbol{q}_i) - f(\boldsymbol{q}_j)\big) \right|, \tag{4}$$

where $sign(\cdot)$ denotes the signum function

$$sign(r) = \begin{cases} 1, & \text{if } r > 0 \\ -1, & \text{if } r \leq 0 \end{cases}.$$

The use of cost functions like Equation (4) leads to intractable optimization problems, therefore, we consider the following least squares approximation, which regresses the differences $s_i - s_j$ with $f(\boldsymbol{q}_i) - f(\boldsymbol{q}_j)$ of the relevant training data points $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$:

$$\mathrm{c}(f, \mathcal{T}) = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij} \Big((s_i - s_j) - (f(\boldsymbol{q}_i) - f(\boldsymbol{q}_j)) \Big)^2. \tag{5}$$

In the above described setting we assume that every instance-label pair has an associated score. A straightforward extension of the proposed algorithm (similar to the one presented in [11]) makes it applicable to the situation when only pairwise preferences are available for the training of the ranker.

4.1 KPCRank

The next theorem characterizes a method we refer to as kernel principal component ranking (KPCRank).

Theorem 1 *Let the training information provided to the algorithm be a tuple $\mathcal{T} = (Q, W, \boldsymbol{s})$. Further, let the projection of the training instance-label pair onto the $l-th$ principal component be $\langle \boldsymbol{v}^l, \Phi(\boldsymbol{q}) \rangle = \frac{1}{\sqrt{m\lambda_l}} \sum_{j=1}^{n} \boldsymbol{a}_j^l k(\boldsymbol{q}_j, \boldsymbol{q})$. Considering the linear ranking model in dimensionality reduced feature space, the prediction function can be written as $f(\boldsymbol{q}) = \sum_{l=1}^{p} \frac{1}{\sqrt{m\lambda_l}} \bar{\boldsymbol{w}}_l \sum_{j=1}^{n} \boldsymbol{a}_j^l k(\boldsymbol{q}_j, \boldsymbol{q})$. Then the coefficient vector $\bar{\boldsymbol{w}}$ for the algorithm*

$$\mathcal{A}(\mathcal{T}) = \operatorname*{argmin}_{f \in \mathcal{H}} J(f),$$

with objective function

$$J(f) = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij} ((s_i - s_j) - (f(\boldsymbol{q}_i) - f(\boldsymbol{q}_j))^2$$

minimizing the least squares approximation of the disagreement error in the dimensionality reduced feature space is

$$\bar{\boldsymbol{w}} = \bar{\Lambda}^{\frac{1}{2}} (\bar{V}^t K L K \bar{V})^{-1} \bar{V}^t K L \boldsymbol{s}, \tag{6}$$

where L is the Laplacian matrix of the graph W.

Proof. We use the fact that for any vector $\boldsymbol{r} \in \mathbb{R}^n$ and an undirected weighted graph W of n vertices, we can write

$$\frac{1}{2}\sum_{i,j=1}^{n} W_{ij}(\boldsymbol{r}_i - \boldsymbol{r}_j)^2 = \boldsymbol{r}^t D\boldsymbol{r} - \boldsymbol{r}^t W\boldsymbol{r} = \boldsymbol{r}^t L\boldsymbol{r},$$

where D and L are the degree matrix and the Laplacian matrix of the graph determined by W. Considering a linear ranking model in the feature space $\mathcal{F}$ the prediction function can be written as $f(Q) = \Phi(Q)\boldsymbol{w}$, where $\boldsymbol{w} \in \mathbb{R}^n$. The objective function in matrix form can be written as

$$J(\boldsymbol{w}) = (\boldsymbol{s} - \Phi(Q)^t\boldsymbol{w})^t L(\boldsymbol{s} - \Phi(Q)^t\boldsymbol{w}). \tag{7}$$

We also assume that our instance-label pairs are centered around zero mean. Therefore, kernel PCA can be performed on the covariance matrix $\Phi(Q)\Phi(Q)^t$ to extract corresponding eigenvectors. Projecting the data onto the principal components we can rewrite Equation (7) as

$$J(\bar{\boldsymbol{w}}) = (\boldsymbol{s} - \Phi(Q)^t V\bar{\boldsymbol{w}})^t L(\boldsymbol{s} - \Phi(Q)^t V\bar{\boldsymbol{w}})$$

or equivalently

$$J(\bar{\boldsymbol{w}}) = (\boldsymbol{s} - K\bar{V}\bar{\Lambda}^{-\frac{1}{2}}\bar{\boldsymbol{w}})^t L(\boldsymbol{s} - K\bar{V}\bar{\Lambda}^{-\frac{1}{2}}\bar{\boldsymbol{w}}).$$

By taking the derivative of $J(\bar{\boldsymbol{w}})$ with respect to $\bar{\boldsymbol{w}}$ we obtain

$$\frac{d}{d\bar{\boldsymbol{w}}}J(\bar{\boldsymbol{w}}) = -2\bar{\Lambda}^{-\frac{1}{2}}\bar{V}^t KL\boldsymbol{s} + 2\bar{\Lambda}^{-\frac{1}{2}}\bar{V}^t KLK\bar{V}\bar{\Lambda}^{-\frac{1}{2}}\bar{\boldsymbol{w}}.$$

We set the derivative to zero and solve with respect to $\bar{\boldsymbol{w}}$

$$\bar{\boldsymbol{w}} = \bar{\Lambda}^{\frac{1}{2}}(\bar{V}^t KLK\bar{V})^{-1}\bar{V}^t KL\boldsymbol{s}. \tag{8}$$

Finally we obtain the predicted score of the unseen instance-label pair based on the first p principal components by

$$f(\boldsymbol{q}) = \sum_{l=1}^{p}\frac{1}{\sqrt{m\lambda_l}}\bar{\boldsymbol{w}}_l\sum_{j=1}^{n}\boldsymbol{a}_j^l k(\boldsymbol{q}_j, \boldsymbol{q}). \tag{9}$$

□

In the next section we demonstrate how to even further simplify Equation (8) and describe ways for efficient multiplication of the matrices involved in the expression. To extract the individual principle components we can use power method described in [6] whose computational complexity scales to $\mathcal{O}(n^2)$ for extraction of individual principal component. The procedure must be repeated for the principal components used for the projection. The computational complexity for making the prediction on a single test data point scales as $\mathcal{O}(pn^2)$ due to the fact that multiplication of matrices KLK can be accomplished efficiently because of the sparseness of L (see Section 4.2) and inversion involved in the Equation (8) is less expensive than multiplication in case $n \gg p$. There are several other strategies that can reduce complexity of the KPCRank algorithm, however, they are outside of the scope of this paper.

4.2 Efficient Selection of Principal Components

Selecting the optimal number of principal components for the KPCRank algorithm can notably improve its performance. There are different ways for efficient extraction of the principal components (e.g. power method) whose complexity scales to $\mathcal{O}(n^2)$ for extracting a single component. Here we describe another method for selecting the optimal number of principal components for the learning task based on a single time performed eigendecomposition of the kernel matrix.

When considering Equation (8) we can observe that by eigendecomposition of the kernel matrix $K = \tilde{V}\tilde{\Lambda}\tilde{V}^t$ and by sorting eigenvalues and corresponding eigenvectors in decreasing order we can further simplify the expression, that is

$$\begin{aligned}\bar{\boldsymbol{w}} &= \bar{\Lambda}^{\frac{1}{2}}(\bar{V}^t K L K \bar{V})^{-1}\bar{V}^t K L \boldsymbol{s} \\ &= (\bar{\Lambda}^{-\frac{1}{2}}\bar{V}^t\tilde{V}\tilde{\Lambda}\tilde{V}^t L\tilde{V}\tilde{\Lambda}\tilde{V}^t\bar{V}\bar{\Lambda}^{-\frac{1}{2}})^{-1}\bar{\Lambda}^{-\frac{1}{2}}\bar{V}^t\tilde{V}\tilde{\Lambda}\tilde{V}^t L\boldsymbol{s} \\ &= (\Lambda^{\frac{1}{2}t}\tilde{V}^t L\tilde{V}\Lambda^{\frac{1}{2}})^{-1}\Lambda^{\frac{1}{2}t}\tilde{V}^t L\boldsymbol{s},\end{aligned}$$

where $\Lambda^{\frac{1}{2}} \in \mathbb{R}^{n\times p}$ is the diagonal matrix containing eigenvalues of K in decreasing order. The latter simplification is possible due to the fact that $\bar{V}^t\tilde{V} = I_{p\times n}$ as well as that both $\bar{\Lambda}^{-\frac{1}{2}}$ and $\tilde{\Lambda}$ are diagonal matrices containing manipulations on eigenvalues of the kernel matrix. The multiplication of the $\Lambda^{\frac{1}{2}t}\tilde{V}^t L\boldsymbol{s}$ can be accomplished in $\mathcal{O}(n^2)$ time since $\boldsymbol{s} \in \mathbb{R}^n$. Furthermore $\Lambda^{\frac{1}{2}t}\tilde{V}^t L\tilde{V}\Lambda^{\frac{1}{2}}$ is a square matrix of dimensions $p\times p$, and in case p is selected to be much smaller than n, the dominating term is multiplication of $\tilde{V}^t L\tilde{V}$, whose computation cost is $\mathcal{O}(un^2)$, where u is the number of instances. This reduction in complexity is achieved due to the sparseness of the Laplacian matrix [17]. Let $M = \Lambda^{\frac{1}{2}t}\tilde{V}^t L\tilde{V}\Lambda^{\frac{1}{2}}$. We can efficiently search for the optimal number of principal components in time $\mathcal{O}(n^2)$. Once we have performed eigendecomposition of the matrix $M = \hat{V}\hat{\Lambda}\hat{V}^t$ (here we assume initial projection of the training data onto all principal components), the subsequent calculation of ${M_p}^{-1}$ based on the p principal components ($p < n$) is computationally inexpensive. It can be performed as follows. When matrix M is decomposed, $\hat{\Lambda} = diag(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix containing the eigenvalues. Set to 0 some of the eigenvalues and leave only p non-zero ones, corresponding to the number of principal components onto which data is projected. Then inverse of matrix M_p can be calculated as

$${M_p}^{-1} = \hat{V}diag(1/\lambda_1, \ldots, 1/\lambda_p, 0, \ldots)\hat{V}^t.$$

Thus, by a single decomposition of the matrix M and subsequent manipulation on eigenvalues we can efficiently search for the optimal number of principal components.

5 Probabilistic Regression on Pairwise Preference Data

In this section we consider learning a ranking function based on pairwise comparison data, that is, data about the ranking function values is provided in terms

of pairwise comparisons at the given locations. Let $D = \{(i_l, j_l, w_l)\}_{l=1:N}$ be a dataset, where N is the number of pairwise comparisons, and $w_l \in \{-1, 1\}$ specifies the direction of the preference. We aim to learn the ranking score function $f(\boldsymbol{x})$. Further, we assume that the observations about the comparisons are noisy and $p(w|f)$ is a Bernoulli distribution, defined as

$$p(w_l|f) = \phi(w_l(f(\boldsymbol{x}_{i_l}) - f(\boldsymbol{x}_{j_l}))),$$

where ϕ is the normal cumulative density function, however, any sigmoid function can be suitable choice. We have chosen the former because it makes some of the computations tractable. We model the function f with a zero mean Gaussian process [9] having covariance function $k(\cdot, \cdot; \boldsymbol{\theta})$. Let the vector $\boldsymbol{f}$ denote the values of the function f in the input variables, that is, the random vector $\boldsymbol{f} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))$. Using Bayes' theorem the Gaussian posterior probability of $\boldsymbol{f}$ can be written as

$$\begin{aligned} p(\boldsymbol{f}|D, \boldsymbol{\theta}) &= \frac{1}{p(D|\boldsymbol{\theta})} \prod_l p(w_l|\boldsymbol{f}) \, p(\boldsymbol{f}|\boldsymbol{\theta}) \\ &= \frac{1}{p(D|\boldsymbol{\theta})} \prod_l \phi(w_l(f(\boldsymbol{x}_{i_l}) - f(\boldsymbol{x}_{j_l}))) \, N(\boldsymbol{f}|\boldsymbol{0}, K), \end{aligned} \tag{10}$$

where K is the covariance matrix with elements $k(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta})$. Since the posterior $p(\boldsymbol{f}|D, \boldsymbol{\theta})$ is analytically intractable we will approximate its mean $\boldsymbol{m}$ and covariance C with the expectation propagation method [10]. Prediction for the ranking function values at a new evaluation point $\boldsymbol{x}_*$ can be computed as

$$\begin{aligned} p(f(\boldsymbol{x}_*)|D, \boldsymbol{\theta}) &= \int p(f(\boldsymbol{x}_*)|\boldsymbol{f}, \boldsymbol{\theta}) \, p(\boldsymbol{f}|D, \boldsymbol{\theta}) \, d\boldsymbol{f} \\ &\approx N\left(f(\boldsymbol{x}_*)|\boldsymbol{k}_*^t K^{-1} \boldsymbol{m}, k_* + \boldsymbol{k}_*^t K^{-1}(C - K) K^{-1} \boldsymbol{k}_*\right), \end{aligned}$$

where $k_* = k(\boldsymbol{x}_*, \boldsymbol{x}_*; \boldsymbol{\theta})$, $K = [k(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta})]_{i,j}$ and $\boldsymbol{k}_* = [k(\boldsymbol{x}_*, \boldsymbol{x}_j; \boldsymbol{\theta})]_i$. Prediction for the comparisons of the ranking function values for locations $\boldsymbol{x}_*^1$ and $\boldsymbol{x}_*^2$ is

$$p\left(f\left(\boldsymbol{x}_*^1\right) > f\left(\boldsymbol{x}_*^2\right)|D, \boldsymbol{\theta}\right) \approx \phi\left(\frac{\left(\boldsymbol{k}_*^1 - \boldsymbol{k}_*^2\right)^T K^{-1} \boldsymbol{m}}{\sqrt{r_{11} + r_{22} - 2r_{12}}}\right), \tag{11}$$

where the 2×2 matrix R is given by

$$R = \left[k(\boldsymbol{x}_*^i, \boldsymbol{x}_*^j, \boldsymbol{\theta})\right]_{i,j=1,2} + \left[\boldsymbol{k}_*^1, \boldsymbol{k}_*^2\right]^t K^{-1}(C - K) K^{-1} \left[\boldsymbol{k}_*^1, \boldsymbol{k}_*^2\right].$$

Since we can approximate the marginal likelihood $p(D|\boldsymbol{\theta})$ using expectation propagation, we carry out maximum likelihood procedure on the hyper-parameters. We use the square exponential covariance function.

Probabilistic counterpart of the RankRLS algorithm would be regression with Gaussian noise and Gaussian processes prior, given the score differences $w_{ij} = s_i - s_j$, that is,

$$p(w_{ij}|f(\boldsymbol{x}_i), f(\boldsymbol{x}_i), v) = N(w_{ij}|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j), 1/v).$$

Then the posterior distribution of the random vector $\boldsymbol{f}$ is

$$p\left(\boldsymbol{f}|D,v,\boldsymbol{\theta}\right)=\frac{1}{p\left(D|v,\boldsymbol{\theta}\right)}\prod_{i,j=1}^{n}N\left(w_{ij}|f\left(\boldsymbol{x}_i\right)-f\left(\boldsymbol{x}_j\right),1/v\right)N\left(\boldsymbol{f}|\mathbf{0},K\right).$$

The posterior distribution $p\left(\boldsymbol{f}|w,v,\boldsymbol{\theta}\right)$ is Gaussian, its mean and covariance matrix can be computed by solving a system of linear equations and inverting a matrix, respectively. Here we choose to optimize the model parameters v and $\boldsymbol{\theta}$ by maximizing the marginal likelihood $p\left(D|v,\boldsymbol{\theta}\right)$. Prediction can be done similar to Equation (11), except that no approximations are needed. Note that predictions obtained by the KPCRank algorithm correspond to the predicted mean values of the Gaussian process regression.

6 Experiments

6.1 Parse Ranking Dataset

We apply the KPCRank algorithm to rank the parses generated from the BioInfer corpus [12] which consists of 1100 manually annotated sentences.[1] The preprocessed data in format of RankSVM [8] is available for download.[2] The parse ranking is similar to the document ranking task frequently encountered in the information retrieval domain. Analogously to the document ranking task, where for each query we have an associated set of retrieved documents, in the parse ranking task for each sentence there is a set of parses that needs to be ranked according to some criteria. A more detailed description of the task is provided in [11]. The main motivation for applying machine learning techniques to the problem of parse ranking is that in many cases a set of built-in heuristics included in the parser software that is used to rank parses is performing not satisfactory and, hence, subsequent ranking or selection methods are needed.

We obtain a scoring for an instance-label pair by comparing the parse to the hand annotated correct parse of its sentence. The feature vectors for instance-label pair are generated using graph kernel (see [11]). The relevant instance-label pairs for the ranking task are those associated with the same instance (see Section 2). All the other pairs are considered to be irrelevant to the task of parse ranking.

Before conducting the experiments we ensure that the data is centered in the feature space. This can be accomplished with the following modifications to the training kernel matrix K_{train} and the test kernel matrix K_{test} [16]

$$\hat{K}_{\text{train}}=(I-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^t)K_{\text{train}}(I-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^t)$$

and

$$\hat{K}_{\text{test}}=(K_{\text{test}}-\frac{1}{n}\mathbf{1}_{n_{\text{test}}}\mathbf{1}_n^tK_{\text{train}})(I-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^t),$$

[1] `www.it.utu.fi/BioInfer`

[2] `www.cs.ru.nl/~evgeni`

where n is number of training data, n_{test} number of test data, $\mathbf{1}_n$ and $\mathbf{1}_{n_{test}}$ are the column vector containing ones of the dimensions $\mathbb{R}^n$ and $\mathbb{R}^{n_{\text{test}}}$ respectively.

In order to select the optimal number of principal components and regularization parameters for the baseline methods, we divide complete dataset of 1100 annotated sentences with the maximum of 3 parses associated with each sentence into two parts. The first part consisting of 500 instance-label pairs is used for training and second part of 2698 instance-label pairs is reserved for final validation. The appropriate values of the regularization and the kernel parameters are determined by grid search with 5-fold cross-validation on the parameter estimation data. The performed experiments can be subdivided into three parts: the experiments on the dataset without noise, experiments on the dataset where scores were intentionally corrupted by Gaussian noise with standard deviation $\sigma = 0.5$ and mean $\mu = 0.0$, and finally with standard deviation $\sigma = 1.0$ and mean $\mu = 0.0$. We note that parse goodness scores present in the dataset are based on the F-score function and, thus, vary between 0 and 1. To avoid influence of the random initialization of the Gaussian noise, we perform the complete experiment 3 times and average the obtained results. The algorithms are trained on the parameter estimation data set with the best found parameter values and tested on instance-label pairs reserved for the final validation. The results of the validation are presented in Table 1.

When no noise is added to the labels of the dataset, the RLS and RankRLS algorithms outperform the KPCRank and KPCR algorithms. Unsatisfactory performance of the KPCRank algorithm can be explained by the fact that by projecting data onto a small number of principal components (in our case less than 100) features that are not most expressive, but that still contribute to the learning performance of the algorithm are lost. Opposite to this, RLS and RankRLS are the methods that do not reduce dimensionality of the feature space, but use regularization controlling the tradeoff between the cost on the training set and the complexity of the hypothesis learned in complete feature space. However, once the noise is added to the dataset KPCRank algorithm performs notably better than the other methods. This is due to the fact that RLS and RankRLS again use complete feature space for learning, but this time inclusion of less expressive noisy features (instance-label pairs) degrades the performance. The normalized version of the disagreement error Equation (4) is used to measure the performance of the ranking algorithms. The error is calculated for each sentence separately and the performance is averaged over all sentences.

6.2 Sinc Dataset

To test performance of the method on pairwise preference data we have constructed simple artificial dataset using *sinc* function as follows. A *sinc* function

$$sinc(x) = \frac{sin(\pi x)}{\pi x},$$

where $x \in [-4, 4]$ is used to generate the values used for creating magnitudes of pairwise preferences. We get 2000 equidistant points from the interval $[-4, 4]$,

Table 1. Comparison of the parse ranking performances of the KPCRank, KPCR, RLS, and RankRLS algorithms using a normalized version of the disagreement error Equation (4) as performance evaluation measure.

Method	Without noise	$\sigma = 0.5$	$\sigma = 1.0$
KPCR	0.40	0.46	0.47
KPCRank	0.37	0.41	0.42
RLS	0.34	0.43	0.46
RankRLS	0.35	0.45	0.47

and sample 1000 for constructing the training pairs and 338 for constructing the test pairs. From these pairs we randomly sample 379 used for the training and 48 for the testing. We consider both models proposed in Section 5, that is, (a) using only pairwise preferences and (b) using pairwise preferences with score differences. The magnitude of pairwise preference is calculated as

$$w = sinc(x) - sinc(x').$$

We have compared probabilistic pairwise regression described in Section 5 to the KPCRank algorithm. To evaluate performance of the methods we use normalized count of incorrectly predicted pairs of data points. The error obtained by using the probabilistic pairwise regression on test dataset is 0.035. The KPCRank algorithm has an error of 0.025, indicating that both methods have a good performance when learning pairwise comparison data.

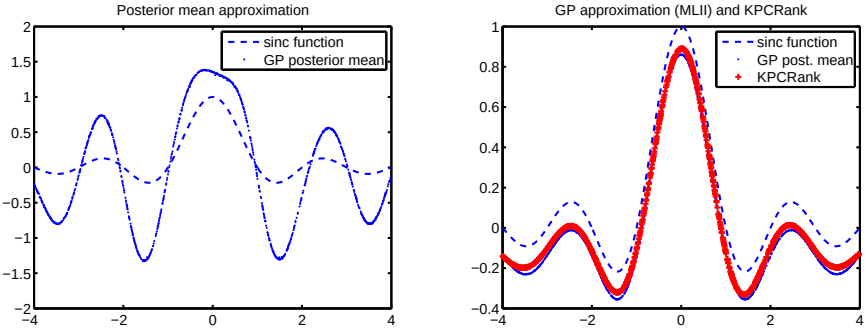

Fig. 1. The *sinc* function and the approximate posterior means of the random vector $\boldsymbol{f}$ using the pairwise preference data and a full Bayesian procedure with a uniform prior on the log of the kernel width parameter (left). The approximation is done with expectation propagation [10]. We have randomly corrupted 5% of the pairwise preferences. That is why the difference between the *sinc* function and the approximate posterior mean $\boldsymbol{f}$ (left) has no influence on the disagreement error as long as the "shape" is correct. The approximate posterior means of the random vector $\boldsymbol{f}$ using the score differences data and the maximum likelihood method on the hype-parameter (right). Note that the model is invariant to the translations of the score function.

7 Conclusions

In this paper we propose the KPCRank algorithm for learning preference relations. The KPCRank algorithm can be considered as an extension of nonlinear principal component regression for learning pairwise preferences. The algorithm belongs to the class of so-called shrinkage methods (similar to KPCR, RLS, etc.) that are designed to shrink the solution from the areas of low data spread and can result in an estimate that is biased but has lower variance. This is accomplished by projecting the training data onto principal components in the feature space, that in turn map onto arbitrary manifold in the input space. Another advantage of the method is that by projecting the data onto the components with higher eigenvalues we aim at discarding the noise contained in the original data and obtain better performance compared to the baseline methods when data is corrupted by noise.

Our experiments confirm that the proposed algorithm works well in situations when the ranking function has to be learned from the noisy data. The KPCRank algorithm notably outperforms the RLS, RankRLS, and KPCR algorithms in the experiments where data is intentionally corrupted by noise. Furthermore, we compare the KPCRank algorithm to the probabilistic regression when learning on pairwise preference data. Our results indicate that both methods achieve good performance by learning correct prediction function.

In the future we are planning to extend the algorithm for learning multiple labels simultaneously, propose methods to further decrease its computational complexity, and test it on various ranking datasets.

Acknowledgments

We acknowledge support from the Netherlands Organization for Scientific Research (NWO), in particular a Vici grant (639.023.604).

References

1. Adriana Birlutiu, Perry Groot, and Tom Heskes. Multi-task Preference learning with Gaussian Processes. In *Proceedings of the 17th European Symposium on Artificial Neural Networks (ESANN)*, pages 123–128, 2009.
2. Wei Chu and Zoubin Ghahramani. Preference learning with gaussian processes. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the Twenty-Second International Conference (ICML 2005)*, volume 119 of *ACM International Conference Proceeding Series*, pages 137–144. ACM, 2005.
3. Michael Collins and Nigel Duffy. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 263–270, Morristown, NJ, USA, 2001. Association for Computational Linguistics.
4. 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.

5. Johannes Fürnkranz and Eyke Hüllermeier. Preference learning. *Künstliche Intelligenz*, 19(1):60–61, 2005.
6. Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.
7. Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17):1897–1916, 2008.
8. Thorsten Joachims. Optimizing search engines using clickthrough data. In David Hand, Daniel Keim, and Raymond Ng, editors, *Proceedings of the 8th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 133–142, New York, NY, USA, 2002. ACM Press.
9. David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
10. Thomas P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT, 2001.
11. 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.
12. Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8(50), 2007.
13. 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.
14. Roman Rosipal and Leonard J. Trejo. Kernel partial least squares regression in reproducing kernel hilbert space. *Journal of Machine Learning Research*, 2:97–123, 2002.
15. Bernhard Schölkopf, Alex J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Artificial Neural Networks - ICANN '97, 7th International Conference*, volume 1327 of *Lecture Notes in Computer Science*, pages 583–588. Springer, 1997.
16. John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.
17. Evgeni Tsivtsivadze, Tapio Pahikkala, Antti Airola, Jorma Boberg, and Tapio Salakoski. A sparse regularized least-squares preference learning algorithm. In Anders Holst, Per Kreuger, and Peter Funk, editors, *10th Scandinavian Conference on Artificial Intelligence (SCAI 2008)*, volume 173, pages 76–83. IOS Press, 2008.