Kernel Principal Component Ranking: Robust Ranking on Noisy Data

Evgeni Tsivtsivadze Botond Cseke Tom Heskes

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

Presentation Outline

Learning on Noisy Data

- Real world data is usually corrupted by noise (e.g. in bioinformatics, natural language processing, information retrieval, etc.)

Learning on Noisy Data

- Real world data is usually corrupted by noise (e.g. in bioinformatics, natural language processing, information retrieval, etc.)
- Learning on noisy data is a challenge: ML methods frequently use low-rank approximation of the data matrix

Learning on Noisy Data

- Real world data is usually corrupted by noise (e.g. in bioinformatics, natural language processing, information retrieval, etc.)
- Learning on noisy data is a challenge: ML methods frequently use low-rank approximation of the data matrix
- Any manifold learner or dimensionality reduction technique can be used for de-noising

Learning on Noisy Data

- Real world data is usually corrupted by noise (e.g. in bioinformatics, natural language processing, information retrieval, etc.)
- Learning on noisy data is a challenge: ML methods frequently use low-rank approximation of the data matrix
- Any manifold learner or dimensionality reduction technique can be used for de-noising
- Our algorithm is an extension of nonlinear principal component regression applicable to preference learning task

Learning to Rank

Learning to rank (total order is given over all data points)

- Applications - collaborative filtering in electronic commerce, protein ranking (e.g. RankProp: Protein Ranking by Network Propagation), parse ranking, etc.
- We aim to learn scoring function that is capable of ranking data points
- Several accepted settings for learning (ref. upcoming Preference Learning Book)
  - Object ranking
  - Label ranking
  - Instance ranking

KPCRank Algorithm

- Main idea: Create new feature space with reduced dimensionality (only most expressive features are preserved) and use the ranking algorithm in that space to learn noise insensitive ranking function

KPCRank Algorithm

- Main idea: Create new feature space with reduced dimensionality (only most expressive features are preserved) and use the ranking algorithm in that space to learn noise insensitive ranking function
- KPCRank scales linearly with the number of data points in the training set and is equal to that of KPCR

KPCRank Algorithm

- Main idea: Create new feature space with reduced dimensionality (only most expressive features are preserved) and use the ranking algorithm in that space to learn noise insensitive ranking function
- KPCRank scales linearly with the number of data points in the training set and is equal to that of KPCR
- KPCRank regularizes by projecting data onto lower dimensional space (number of principal components is a model parameter)

KPCRank Algorithm

- Main idea: Create new feature space with reduced dimensionality (only most expressive features are preserved) and use the ranking algorithm in that space to learn noise insensitive ranking function
- KPCRank scales linearly with the number of data points in the training set and is equal to that of KPCR
- KPCRank regularizes by projecting data onto lower dimensional space (number of principal components is a model parameter)
- In conducted experiments KPCRank performs better than the baseline methods when learning to rank from data corrupted by noise

Dimensionality Reduction

Consider covariance matrix

$$C = \frac{1}{m}\sum_{i=1}^{m}\Phi(z_i)\Phi(z_i)^t = \frac{1}{m}\Phi(Z)\Phi(Z)^t$$

To find the first principal component we solve

$$Cv = \lambda v$$

The key observation: $v = \sum_{i=1}^{m} a_i\Phi(z_i)$, therefore,

$$\frac{1}{m}Ka = \lambda a$$

$$\langle v^l, \Phi(z)\rangle = \frac{1}{\sqrt{m\lambda_l}}\sum_{i=1}^{m} a_i^l\langle\Phi(z_i)\Phi(z)\rangle = \frac{1}{\sqrt{m\lambda_l}}\sum_{i=1}^{m} a_i^l k(z_i, z)$$

KPCRank Algorithm

We start with the disagreement error:

$$d(f, T) = \frac{1}{2} \sum_{i,j=1}^{m} W_{ij} \left| \operatorname{sign}\left(s_i - s_j\right) - \operatorname{sign}\left(f(z_i) - f(z_j)\right) \right|.$$

The least squares ranking objective is

$$J(w) = (S - \Phi(Z)^t w)^t L (S - \Phi(Z)^t w)$$

and using projected data (reduced feature space) the objective can be rewritten as

$$J(\bar{w}) = (S - \Phi(Z)^t V \bar{w})^t L (S - \Phi(Z)^t V \bar{w})$$

Regularization is performed by selecting optimal number of principle components.

KPCRank Algorithm

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

$$\bar{w} = \bar{\Lambda}^{\frac{1}{2}}(\bar{V}^t KLK\bar{V})^{-1}\bar{V}^t KLS$$

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

$$f(z) = \sum_{l=1}^{p} \frac{1}{\sqrt{m\lambda_l}} \bar{w}_l \sum_{j=1}^{m} a_j^l k(z_j, z)$$

- Efficient selection of the optimal number of principal components
- Detailed computation complexity considerations
- Alternative approaches for reducing computational complexity (e.g. subset method)

Experiments

- Label ranking - Parse Ranking dataset
- Pairwise preference learning - Synthetic dataset based on sinc(x) function
- Baseline methods: Regularized least-squares, RankRLS, KPC regression, Probabilistic ranker.

Parse Ranking Dataset

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

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

A Probabilistic Ranker

A probabilistic counterpart of the RankRLS algorithm would be regression with Gaussian noise and Gaussian processes prior. Given the score differences $w_{ij} = s_i - s_j$

$$p\left(w_{ij}|f\left(x_i\right), f\left(x_i\right), v\right) = N\left(w_{ij}|f\left(x_i\right) - f\left(x_j\right), 1/v\right).$$

Then the posterior distribution is

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

- The posterior distribution $p\left(f|w, v, \boldsymbol{\theta}\right)$ is Gaussian, its mean and covariance matrix can be computed by solving a system of linear equations and inverting a matrix, respectively.
- Note that predictions obtained by the RankRLS algorithm correspond to the predicted mean values of the Gaussian process regression

Sinc Dataset

We use *sinc* function

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

to generate the values used for creating magnitudes of pairwise preferences.

- We get 2000 equidistant points from the interval $[-4, 4]$
- 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

The magnitude of pairwise preference is calculated as

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

Sinc Dataset

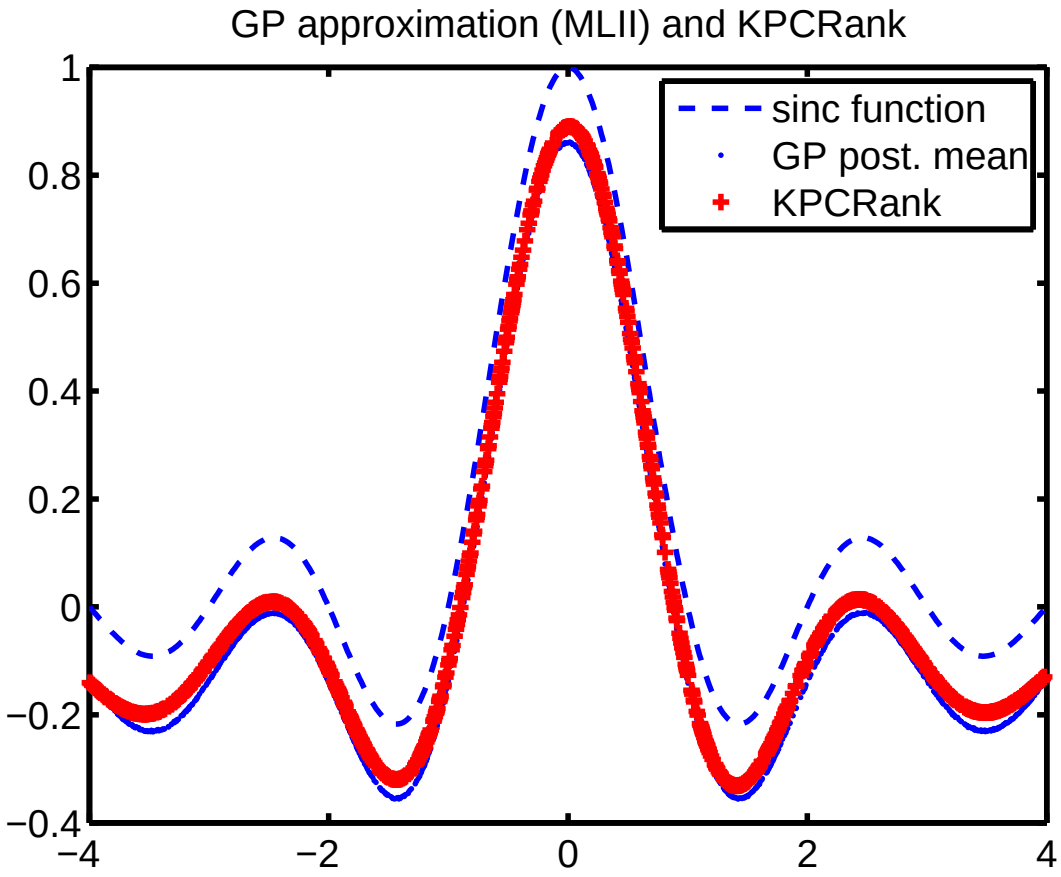

Figure: The *sinc* function and the approximate posterior means of the f using the preference with magnitudes and KPCRank predictions

Thank you.