Greedy RankRLS: a Linear Time Algorithm for Learning Sparse Ranking Models

Tapio Pahikkala Antti Airola Pekka Naula Tapio Salakoski

Turku Centre for Computer Science (TUCS),
Department of Information Technology, University of Turku,
Joukahaisenkatu 3-5 B, 20520 Turku, Finland
{firstname.lastname}@it.utu.fi

August 1, 2010

Feature selection for RankRLS

- We introduce greedy RankRLS, a feature selection algorithm for RankRLS whose time complexity scales linearly in the number of features to be selected, the overall number of features, and the number of training examples.
- Greedy RankRLS produces ranking models that are exactly equivalent with those obtained with the standard wrapper approach for RankRLS with greedy forward selection and leave-query-out cross-validation as a model selection criterion.
- The proposed algorithm is shown to work well in experiments with LETOR data set.

Wrapper type of feature selection

Wrapper type of feature selection methods select features through interaction with a learning algorithm which is used as a black-box method. Simply put, the wrapper technique requires the following components:

- Base learning algorithm around which the feature selection algorithm is wrapped.
- Search strategy over the power set of features.
- Heuristic for assessing the goodness of the feature subsets.

Wrapper type of feature selection

- As a base learner we have RankRLS, a simple algorithm for learning to rank which is based on a modification of regularized least-squares for ranking tasks.
- As a search strategy we use greedy forward selection which starts from an empty feature set and on each iteration the feature, whose addition yields the best value of the selection heuristic, is selected.
- As a selection criterion we use leave-query-out (LQO) cross-validation. The LQO cross-validation can be used together with ranking performance measures.

Linear RankRLS

Similarly to many other learning to rank algorithms, RankRLS minimizes a pairwise loss function plus a regularization term:

$$\underset{\mathbf{w}\in\mathbb{R}^{|\mathcal{S}|}}{\operatorname{argmin}} \left\{ \sum_{\mathcal{Q}\in Q} \frac{1}{2|\mathcal{Q}|} \sum_{i,j\in\mathcal{Q}} (\mathbf{y}_i - \mathbf{y}_j - \mathbf{w}^\mathsf{T}\mathbf{X}_{\mathcal{S},i} + \mathbf{w}^\mathsf{T}\mathbf{X}_{\mathcal{S},j})^2 + \lambda\|\mathbf{w}\|^2 \right\}$$

Notation

$\mathbf{X}$	Training data matrix with n features and m data points.
$\mathbf{y}$	Label vector.
λ	Regularization parameter.
Q	Partition of training example indices according to queries.
$\mathcal{S}$	Index set of selected features.

Motivation for using L_2 loss for ranking

- The ranking performance of RankRLS is essentially the same as that of RankSVM.
- Performance evaluation time scales linearly with respect to the number of data points
- Training time scales linearly with respect to the number of training examples
- RankRLS has a simple closed form solution, which can be fully expressed in terms of matrix operations.
- Efficient computational short-cuts for cross-validation and for adding new features as well as for their combination.

Pairwise squared error via query-wise centering

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}^1 & & \\ & \ddots & \\ & & \mathbf{L}^q \end{pmatrix}, \qquad \mathbf{L}^i = \mathbf{I}^{|\mathcal{Q}_i| \times |\mathcal{Q}_i|} - \frac{1}{|\mathcal{Q}_i|}\mathbf{1}\mathbf{1}^\mathsf{T}$$

$$\sum_{\mathcal{Q} \in Q} \frac{1}{2|\mathcal{Q}|} \sum_{i,j \in \mathcal{Q}} (\mathbf{e}_i - \mathbf{e}_j)^2 = \mathbf{e}^\mathsf{T}\mathbf{L}\mathbf{e}$$

Notation

$\mathbf{X}$	$n \times m$ training data matrix with n features and m data points.
$\mathbf{y}$	Label vector.
λ	Regularization parameter.
$\mathbf{I}^{\lvert\mathcal{Q}_i\rvert \times \lvert\mathcal{Q}_i\rvert}$	Identity matrix of size $\lvert\mathcal{Q}_i\rvert \times \lvert\mathcal{Q}_i\rvert$.
$\mathbf{1}$	Vector of ones of size $\lvert\mathcal{Q}_i\rvert$.

Pairwise squared error via query-wise centering

- With query-wise centering, the pairwise squared error can be computed in linear time with respect to the number of data points, because of the sparse decomposition of $\mathbf{L}$.
- Works with arbitrary relevance levels and with partitions of data into queries
- Straighforward and fast to optimize.
- The matrix $\mathbf{L}$ is idempotent, which eases algorithm analysis.
- Allows reformulating RankRLS as standard RLS.

Reformulation of RankRLS as RLS

$$\underset{\mathbf{w}\in\mathbb{R}^{|\mathcal{S}|}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{m} (\widehat{\mathbf{y}}_i - \mathbf{w}^{\mathsf{T}} \widehat{\mathbf{X}}_{\mathcal{S},i})^2 + \lambda \|\mathbf{w}\|^2 \right\}$$

$$\begin{aligned} \widehat{\mathbf{X}} &:= \mathbf{XL} \\ \widehat{\mathbf{y}} &:= \mathbf{Ly} \end{aligned}$$

Notation

- $\widehat{\mathbf{X}}$ Query-wise centered training data matrix.
- $\widehat{\mathbf{y}}$ Query-wise centered Label vector.
- λ Regularization parameter.
- $\mathcal{S}$ Index set of selected features.

Leave-query-out cross-validation

$$\mathrm{LQO}(\mathbf{X}_{\mathcal{S}}, \mathbf{y}, Q, \lambda) = \frac{1}{|Q|} \sum_{\mathcal{Q} \in Q} l(\mathbf{w}^{\mathcal{Q}^{\mathsf{T}}} \mathbf{X}_{\mathcal{S},\mathcal{Q}}, \mathbf{y}_{\mathcal{Q}})$$

where

$$\mathbf{w}^{\mathcal{Q}} = \mathrm{RankRLS}(\mathbf{X}_{\mathcal{S},\mathcal{I}\setminus\mathcal{Q}}, \mathbf{y}_{\mathcal{I}\setminus\mathcal{Q}}, Q \setminus \{\mathcal{Q}\}, \lambda)$$

Notation

$\mathbf{X}$	Training data matrix with n features and m data points.
$\mathbf{y}$	Label vector.
k	The desired number of features to be selected.
λ	Regularization parameter.
Q	Partition of training example indices according to queries.
$\mathcal{S}$	Index set of selected features.
RankRLS	The RankRLS training algorithm.
l	Loss function or performance measure.

Leave-query-out cross-validation

- Maximal use of training data, that is, all but one query is used for training in each cross-validation round.
- Almost unbiased estimator of the ranking performance.
- Guarantees that data points related to the same query are never split between the training and test folds.
- Straightforward to combine with ranking performance measures which are computed for each query separately.
- Obtaining LQO performance is computationally efficient for RankRLS due to short-cuts based on matrix algebra.

Greedy forward selection

Input: $\mathbf{X}$, $\mathbf{y}$, Q, k, λ
Output: $\mathcal{S}$, $\mathbf{w}$
$\mathcal{S} \leftarrow \emptyset$;
while $|\mathcal{S}| < k$ **do**
  $b \leftarrow \text{argmin}_{i \in \{1,\ldots,n\} \setminus \mathcal{S}} \text{LQO}(\mathbf{X}_{\mathcal{S} \cup \{i\}}, \mathbf{y}, Q, \lambda)$;
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{b\}$;
$\mathbf{w} \leftarrow \text{RankRLS}(\mathbf{X}_{\mathcal{S}}, \mathbf{y}, Q, \lambda)$;

Notation

$\mathbf{X}$	Training data matrix with n features and m data points.
$\mathbf{y}$	Label vector.
k	The desired number of features to be selected.
λ	Regularization parameter.
Q	Partition of training example indices according to queries.
LQO	Leave-query-out cross-validation error for RankRLS.
RankRLS	RankRLS training algorithm.

Computational complexity considerations

A straightforward implementation of the wrapper type of feature selection for RankRLS requires $O(\min\{k^3mnq, k^2m^2nq\})$ time, because:

- Learning a linear RLS predictor with k features and m training examples requires $O(\min\{k^2m, km^2\})$ time.
- The greedy forward selection has k iterations if k features are to be selected.
- The greedy forward selection goes through of the order of $O(n)$ features available for selection in each iteration.
- LQO heuristic has q iterations.

Notation

k	Number of features to be selected.
m	Number of training examples.
n	Overall number of available features.
q	Number of queries in the training set.

Computational complexity considerations

Greedy RankRLS, our novel algorithmic implementation of the wrapper type of feature selection for RankRLS, requires only $O(kmn)$ time and $O(mn)$ space, while it provides results that are exactly equivalent with the wrapper technique.

- Computing the LQO predictions for the m training examples can be done in $O(m)$ time
- The pairwise squared ranking performance can be computed from the LQO predictions in $O(m)$ time due to the centering trick
- The LQO predictions are separately computed for $O(n)$ features available for addition in each round of greedy RankRLS
- Greedy RankRLS has k rounds

Notation

k	Number of features to be selected.
m	Number of training examples.
n	Overall number of available features.

Input: $\widehat{\mathbf{X}}$, $\widehat{\mathbf{y}}$, Q, k, λ
Output: $\mathcal{S}$, $\mathbf{w}$

$\mathbf{a} \leftarrow \lambda^{-1}\widehat{\mathbf{y}}$; $\quad \mathbf{C} \leftarrow \lambda^{-1}\widehat{\mathbf{X}}^{\mathsf{T}}$; $\quad \mathbf{U} \leftarrow \widehat{\mathbf{X}}^{\mathsf{T}}$; $\quad \mathbf{p} \leftarrow \widehat{\mathbf{y}}$; $\quad \mathcal{S} \leftarrow \emptyset$;

while $|\mathcal{S}| < k$ **do**

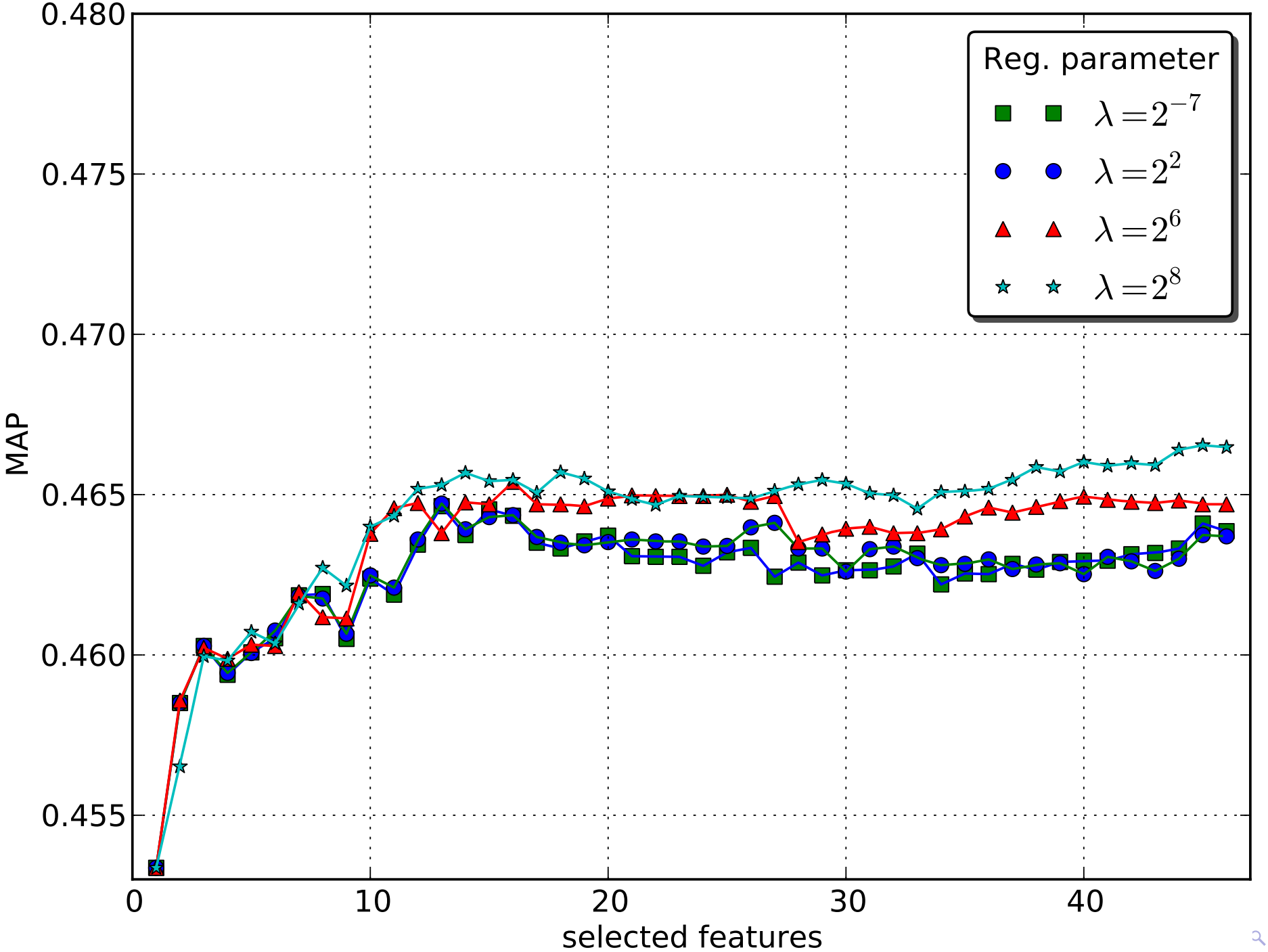
- $e \leftarrow \infty$;
- $b \leftarrow 0$;
- **foreach** $i \in \{1, \ldots, n\} \setminus \mathcal{S}$ **do**
  - $c \leftarrow (1 + \widehat{\mathbf{X}}_i \mathbf{C}_{:,i})^{-1}$;
  - $d \leftarrow c\mathbf{C}^{\mathsf{T}}{}_i \widehat{\mathbf{y}}$;
  - $e_i \leftarrow 0$;
  - **foreach** $\mathcal{Q} \in Q$ **do**
    - $\gamma \leftarrow (-c^{-1} + \mathbf{C}^{\mathsf{T}}{}_{i,\mathcal{Q}} \mathbf{U}_{\mathcal{Q},i})^{-1}$;
    - $\tilde{\mathbf{p}}_{\mathcal{Q}} \leftarrow \mathbf{p}_{\mathcal{Q}} - d\mathbf{U}_{\mathcal{Q},i} - \gamma \mathbf{U}_{\mathcal{Q},i}(\mathbf{U}^{\mathsf{T}}{}_{i,\mathcal{Q}}(\mathbf{a}_{\mathcal{Q}} - d\mathbf{C}_{\mathcal{Q},i}))$;
    - $e_i \leftarrow e_i + (\tilde{\mathbf{p}}_{\mathcal{Q}})^{\mathsf{T}} \tilde{\mathbf{p}}_{\mathcal{Q}}$;
  - **if** $e_i < e$ **then**
    - $e \leftarrow e_i$;
    - $b \leftarrow i$;
- $c \leftarrow (1 + \widehat{\mathbf{X}}_b \mathbf{C}_{:,b})^{-1}$;
- $d \leftarrow c\mathbf{C}^{\mathsf{T}}{}_b \widehat{\mathbf{y}}$;
- $\mathbf{t} \leftarrow c\widehat{\mathbf{X}}_b \mathbf{C}$;
- **foreach** $\mathcal{Q} \in Q$ **do**
  - $\gamma \leftarrow (-c^{-1} + \mathbf{C}^{\mathsf{T}}{}_{b,\mathcal{Q}} \mathbf{U}_{\mathcal{Q},b})^{-1}$;
  - $\mathbf{p}_{\mathcal{Q}} \leftarrow \mathbf{p}_{\mathcal{Q}} - d\mathbf{U}_{\mathcal{Q},b} - \gamma \mathbf{U}_{\mathcal{Q},b}(\mathbf{U}^{\mathsf{T}}{}_{b,\mathcal{Q}}(\mathbf{a}_{\mathcal{Q}} - d\mathbf{C}_{\mathcal{Q},b}))$;
  - $\mathbf{U}_{\mathcal{Q}} \leftarrow \mathbf{U}_{\mathcal{Q}} - \mathbf{U}_{\mathcal{Q},b}\mathbf{t} - \gamma \mathbf{U}_{\mathcal{Q},b}(\mathbf{U}^{\mathsf{T}}{}_{b,\mathcal{Q}}(\mathbf{C}_{\mathcal{Q}} - \mathbf{C}_{\mathcal{Q},b}\mathbf{t}))$;
- $\mathbf{a} \leftarrow \mathbf{a} - d\mathbf{C}_{:,b}$;
- $\mathbf{C} \leftarrow \mathbf{C} - \mathbf{C}_{:,b}\mathbf{t}$;
- $\mathcal{S} \leftarrow \mathcal{S} \cup \{b\}$;

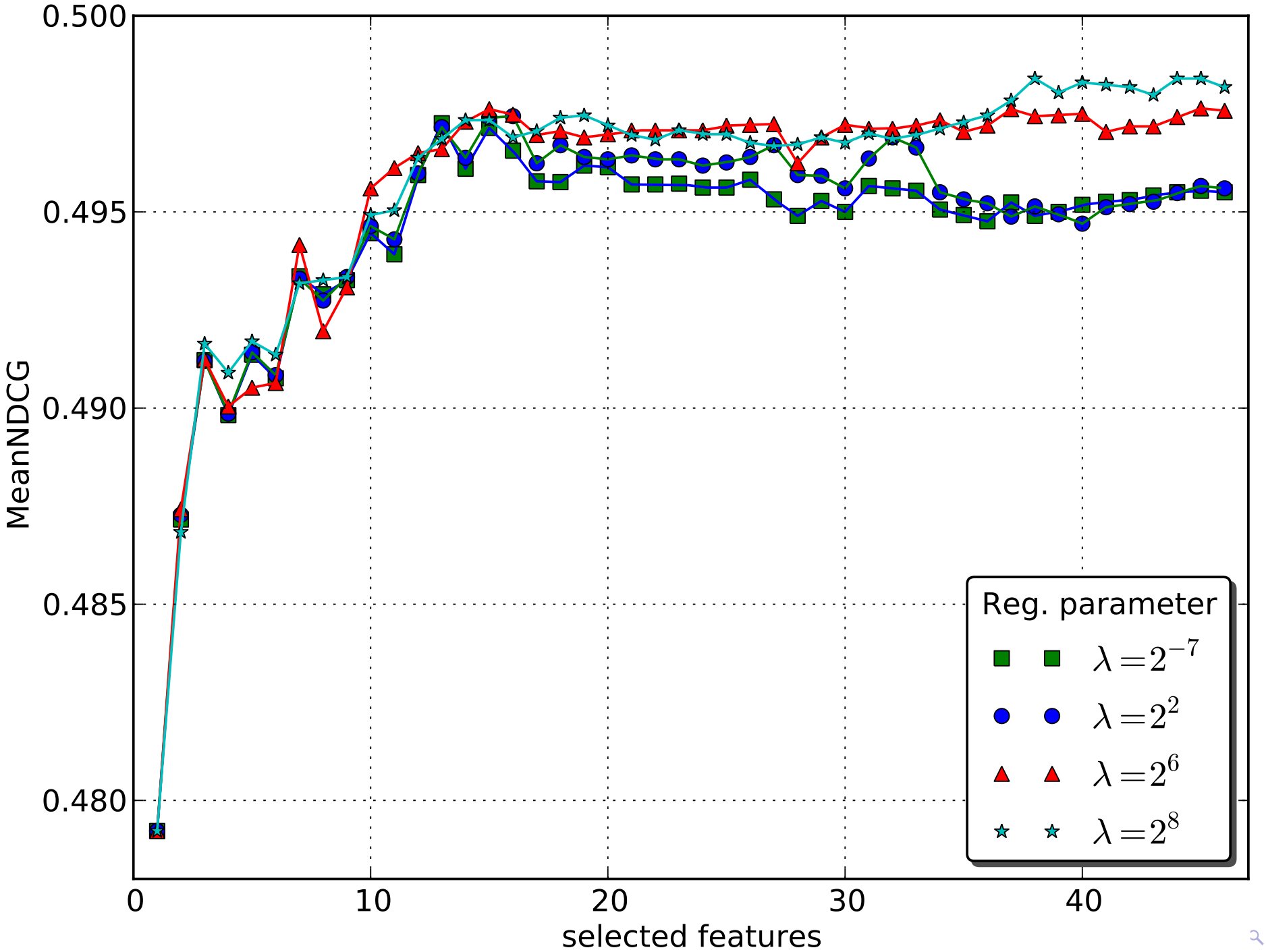
$\mathbf{w} \leftarrow \widehat{\mathbf{X}}_{\mathcal{S}} \mathbf{a}$;

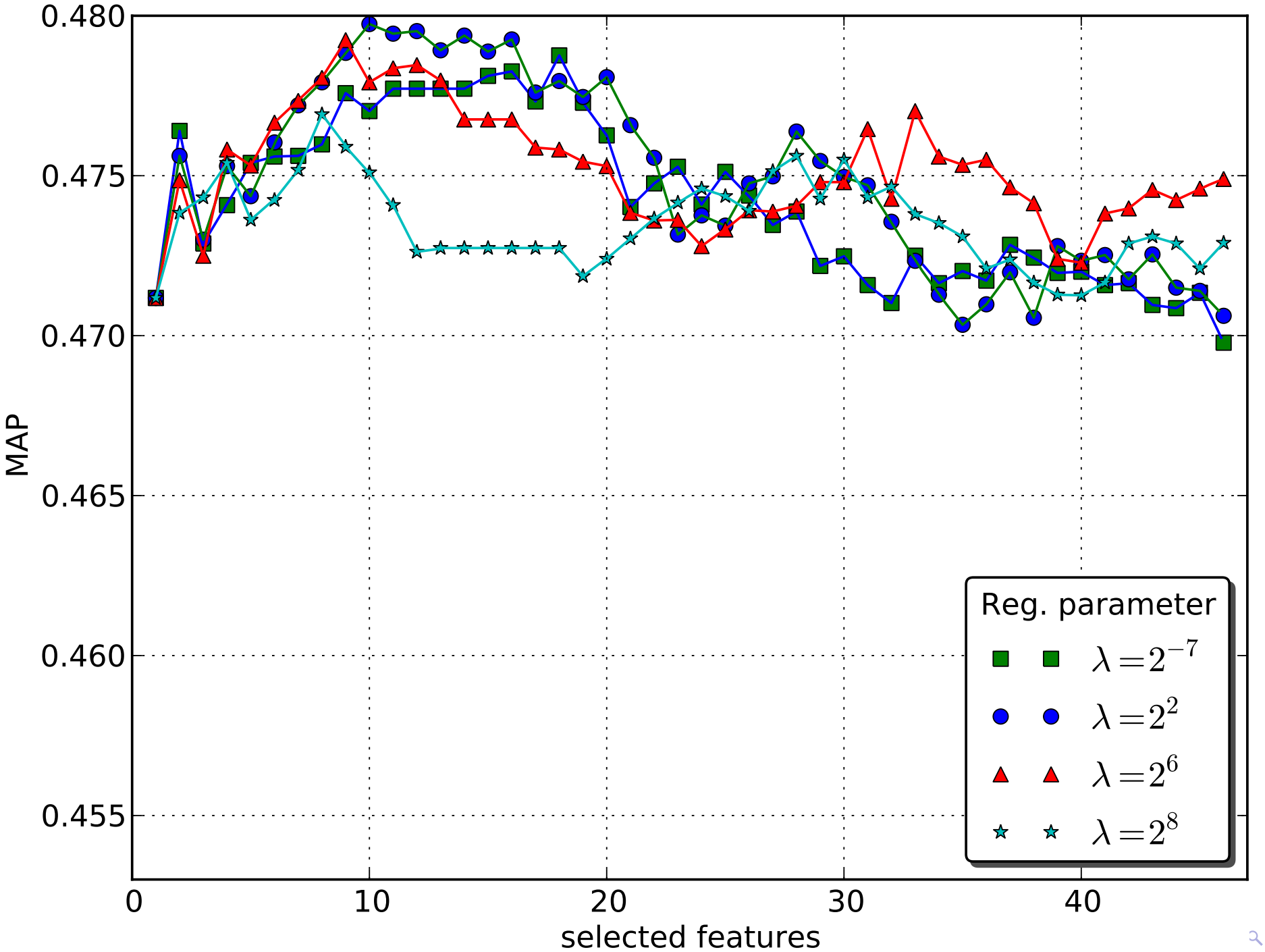
Experiments

We perform experiments on the publicly available LETOR benchmark data set (version 4.0) for learning to rank for information retrieval
`http://research.microsoft.com/en-us/um/beijing/projects/letor/`
In particular, we run experiments on the MQ2007 and MQ2008 data sets.

- MQ2007 consists of 69623 examples divided into 1700 queries.
- MQ2008 contains 15211 examples divided into 800 queries.
- The examples in both data sets have 46 high-level features.
- The experimental setup proposed by the authors of LETOR is followed.
- The value of the regularization parameter λ and the number of features to be selected k are chosen according to the validation results.
- RankRLS and RankSVM are used as baselines.


0.480
0.475
0.470
0.465
0.460
0.455
MAP
0
10
20
30
40
selected features
Reg. parameter
$\lambda = 2^{-7}$
$\lambda = 2^{2}$
$\lambda = 2^{6}$
$\lambda = 2^{8}$


0.500
0.495
0.490
0.485
0.480
MeanNDCG
0
10
20
30
40
selected features
Reg. parameter
$\lambda = 2^{-7}$
$\lambda = 2^{2}$
$\lambda = 2^{6}$
$\lambda = 2^{8}$


0.480
0.475
0.470
0.465
0.460
0.455
MAP
0
10
20
30
40
selected features
Reg. parameter
$\lambda = 2^{-7}$
$\lambda = 2^{2}$
$\lambda = 2^{6}$
$\lambda = 2^{8}$

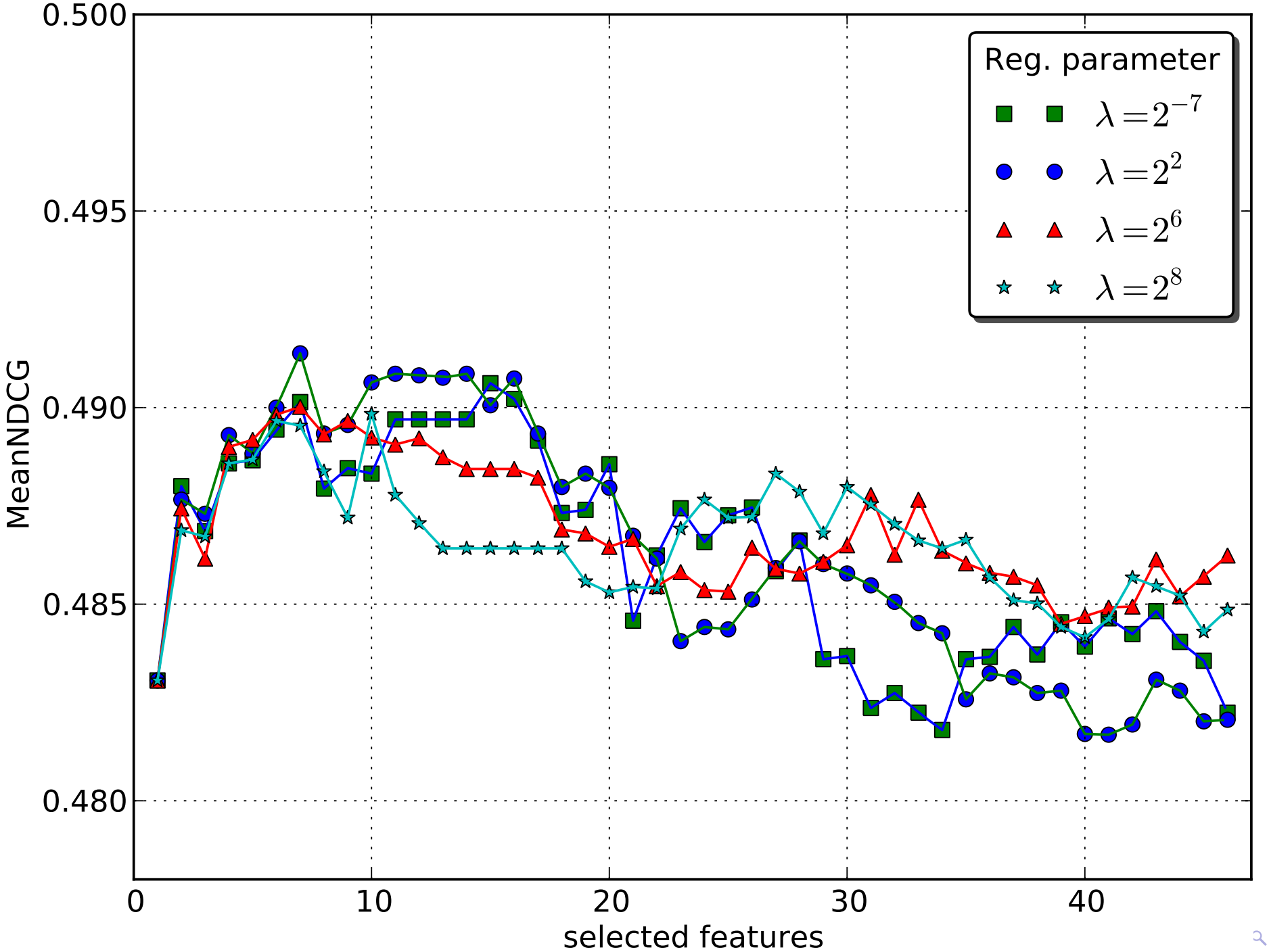

0.500
0.495
0.490
0.485
0.480
MeanNDCG
0
10
20
30
40
selected features
Reg. parameter
$\lambda = 2^{-7}$
$\lambda = 2^{2}$
$\lambda = 2^{6}$
$\lambda = 2^{8}$

Table: Selected features on MQ2007.

Model	fold1	fold2	fold3	fold4	fold5
λ	2^8	2^6	2^9	2^8	2^7
k	11	40	46	44	12
selected 1	39	39	39	39	39
selected 2	19	32	27	28	25
selected 3	25	19	23	45	19
selected 4	23	26	19	23	43
selected 5	32	23	13	43	23
selected 6	16	16	18	33	29
selected 7	43	5	42	13	22
selected 8	22	33	33	18	18
selected 9	5	18	16	22	5
selected 10	33	3	5	15	16

Feature number 39: LMIR.DIR of whole document

Table: Selected features on MQ2008

Model	fold1	fold2	fold3	fold4	fold5
λ	2^0	2^{10}	2^3	2^6	2^0
k	1	4	7	4	1
selected 1	39	39	39	39	39
selected 2		23	29	29	
selected 3		37	25	25	
selected 4		32	23	23	
selected 5			46		
selected 6			37		
selected 7			19		

Feature number 39: LMIR.DIR of whole document

Table: MeanNDCG results on MQ2007

Fold	GRankRLS	RankRLS	RankSVM
1	0.5228	0.5281	0.5278
2	0.4840	0.4841	0.4810
3	0.5056	0.5056	0.5042
4	0.4757	0.4754	0.4699
5	0.5033	0.5003	0.5003
avg	0.4983	0.4987	0.4966

Table: MeanNDCG results on MQ2008

Fold	GRankRLS	RankRLS	RankSVM
1	0.4454	0.4633	0.4577
2	0.4186	0.4269	0.4296
3	0.4787	0.4741	0.4686
4	0.5403	0.5407	0.5442
5	0.5369	0.5138	0.5159
avg	0.4840	0.4838	0.4832

RLScore software

RankRLS and greedy RankRLS, as well as our other previously proposed machine learning algorithms, will be implemented as part of the RLScore open source machine learning framework.

Homepage

`www.tucs.fi/RLScore`

Conclusions

We introduce greedy RankRLS, an algorithm for learning sparse ranking models which

- has RankRLS as a base learning algorithm
- uses greedy forward selection as a search strategy in the power set of features
- uses leave-query-out as a selection heuristic
- has computational complexity $O(kmn)$ (linear in the number of features to be selected, the overall number of features, and the number of training examples)
- performs well in practical experiments