One-Pass Ranking Models for Low-Latency Product Recommendations

Martin Saveski
@msaveski

MIT
(Amazon Berlin)

One-Pass Ranking Models for Low-Latency Product Recommendations

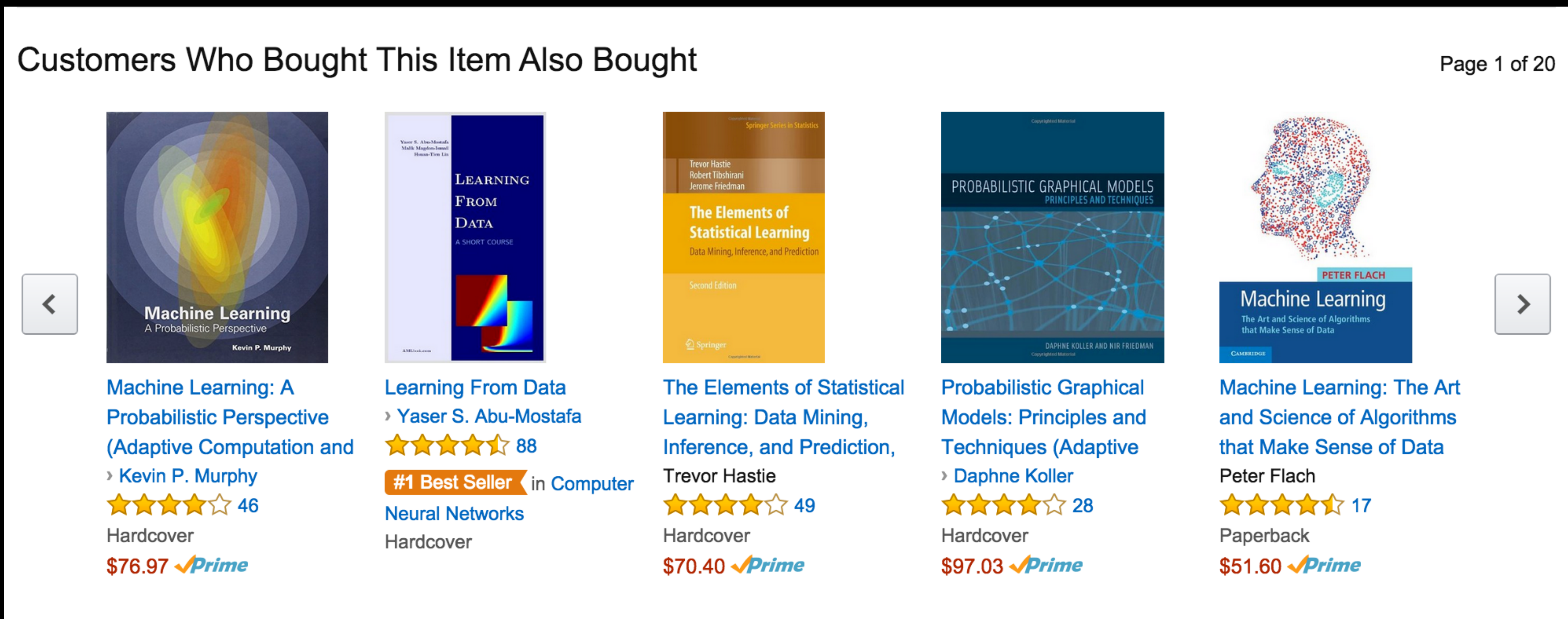
Amazon Machine Learning Team, Berlin

Antonino Freno

Rodolphe Jenatton

Cédric Archambeau

Product Recommendations



Product Recommendations

Constraints

Product Recommendations

Constraints

1. Large # of examples
   Large # of features

Product Recommendations

Constraints

1. Large # of examples
   Large # of features

2. Drifting distribution

Product Recommendations

Constraints

1. Large # of examples
   Large # of features

2. Drifting distribution

3. Real-time ranking
   (<few ms)

Product Recommendations

Constraints

1. Large # of examples
   Large # of features → Small memory footprint

2. Drifting distribution

3. Real-time ranking
   (<few ms)

Product Recommendations

Constraints

1. Large # of examples
   Large # of features ⟶ Small memory footprint

2. Drifting distribution ⟶ Fast training time

3. Real-time ranking
   (<few ms)

Product Recommendations

Constraints

1. Large # of examples
   Large # of features → Small memory footprint
2. Drifting distribution → Fast training time
3. Real-time ranking (<few ms) → Low prediction latency

Our approach

Product Recommendations

Small memory footprint

Fast training time

Low prediction latency

Our approach
Product Recommendations

Small memory footprint

Fast training time

Low prediction latency

Stochastic optimization

One pass learning

Our approach

Product Recommendations

Small memory footprint

Fast training time

→ Stochastic optimization

One pass learning

Low prediction latency → Sparse models

Learning Ranking Functions

Learning Ranking Functions

Three broad families of models

1. Pointwise (Logistic regression)
2. Pairwise (RankSVM)
3. Listwise (ListNet)

Learning Ranking Functions

Three broad families of models

1. Pointwise (Logistic regression)
2. Pairwise (RankSVM)
3. Listwise (ListNet)

Loss functions

- Evaluation functions (NDCG)
- Surrogate functions

Loss Function

Lambda Rank (Burges et al., 2007)

Loss Function

Lambda Rank (Burges et al., 2007)

	Product 1	Product 2	Product 3	Product 4
$\mathbf{X}$: Features	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$
$\mathbf{r}$: Ground-truth Rank	1	1	2	3

Loss Function

Lambda Rank (Burges et al., 2007)

	Product 1	Product 2	Product 3	Product 4
$\mathbf{X}$: Features	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$
$\mathbf{r}$: Ground-truth Rank	1	1	2	3
		i		j

Loss Function

Lambda Rank (Burges et al., 2007)

	Product 1	Product 2	Product 3	Product 4
$\mathbf{X}$: Features	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$
$\mathbf{r}$: Ground-truth Rank	1	1	2	3
		i		j

Importance of sorting i and j correctly

$$\Delta\mathcal{M} = \mathcal{M}(\mathbf{r}) - \mathcal{M}(\mathbf{r}_{i/j})$$

Loss Function

Lambda Rank (Burges et al., 2007)

	Product 1	Product 2	Product 3	Product 4
$\mathbf{X}$: Features	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$
$\mathbf{r}$: Ground-truth Rank	1	1	2	3
		i		j

Importance of sorting i and j correctly

$$\Delta\mathcal{M} = \mathcal{M}(\mathbf{r}) - \mathcal{M}(\mathbf{r}_{i/j})$$

Difference in scores

$$\Delta S = \max\{0, \mathbf{w^T x_j} - \mathbf{w^T x_i}\}$$

Loss Function

Lambda Rank (Burges et al., 2007)

	Product 1	Product 2	Product 3	Product 4
$\mathbf{X}$: Features	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$
$\mathbf{r}$: Ground-truth Rank	1	1	2	3
		i		j

Importance of sorting i and j correctly

$$\Delta\mathcal{M} = \mathcal{M}(\mathbf{r}) - \mathcal{M}(\mathbf{r}_{i/j})$$

Difference in scores

$$\Delta S = \max\{0, \mathbf{w}^{\mathbf{T}}\mathbf{x_j} - \mathbf{w}^{\mathbf{T}}\mathbf{x_i}\}$$

Loss

$$L(\mathbf{X}; \mathbf{w}) = \sum_{\mathbf{r}_i \leq \mathbf{r}_j} \Delta\mathcal{M} \cdot \Delta S$$

ElasticRank

Introducing Sparsity

Adding l_1 and l_2 penalties

$$L^*(\mathbf{X}, \mathbf{w}) = L(\mathbf{X}, \mathbf{w}) + \lambda_1 ||\mathbf{w}||_1 + \frac{1}{2}\lambda_2 ||\mathbf{w}||_2^2$$

ElasticRank

Introducing Sparsity

Adding l_1 and l_2 penalties

$$L^*(\mathbf{X}, \mathbf{w}) = L(\mathbf{X}, \mathbf{w}) + \lambda_1 ||\mathbf{w}||_1 + \frac{1}{2}\lambda_2 ||\mathbf{w}||_2^2$$

Both λ_1 and λ_2 control model complexity

ElasticRank

Introducing Sparsity

Adding l_1 and l_2 penalties

$$L^*(\mathbf{X}, \mathbf{w}) = L(\mathbf{X}, \mathbf{w}) + \lambda_1 ||\mathbf{w}||_1 + \frac{1}{2}\lambda_2 ||\mathbf{w}||_2^2$$

Both λ_1 and λ_2 control model complexity

- λ_1 trades-off sparsity and performance

ElasticRank

Introducing Sparsity

Adding l_1 and l_2 penalties

$$L^*(\mathbf{X}, \mathbf{w}) = L(\mathbf{X}, \mathbf{w}) + \lambda_1 ||\mathbf{w}||_1 + \frac{1}{2}\lambda_2 ||\mathbf{w}||_2^2$$

Both λ_1 and λ_2 control model complexity

- λ_1 trades-off sparsity and performance
- λ_2 adds strong convexity & improves convergence

Optimization Algorithms

Extensions of Stochastic Gradient Descent

Optimization Algorithms

Extensions of Stochastic Gradient Descent

FOBOS Forward-Backward Splitting (Duchi, 2009)

1. Gradient step
2. Proximal step involving the regularization

Optimization Algorithms

Extensions of Stochastic Gradient Descent

FOBOS Forward-Backward Splitting (Duchi, 2009)

1. Gradient step
2. Proximal step involving the regularization

RDA Regularized Dual Averaging (Xiao, 2010)

- Keeps a running average of all past gradients
- Solves a proximal step using the average

Optimization Algorithms

Extensions of Stochastic Gradient Descent

FOBOS Forward-Backward Splitting (Duchi, 2009)

1. Gradient step
2. Proximal step involving the regularization

RDA Regularized Dual Averaging (Xiao, 2010)

- Keeps a running average of all past gradients
- Solves a proximal step using the average

pSGD Pruned Stochastic Gradient Descent

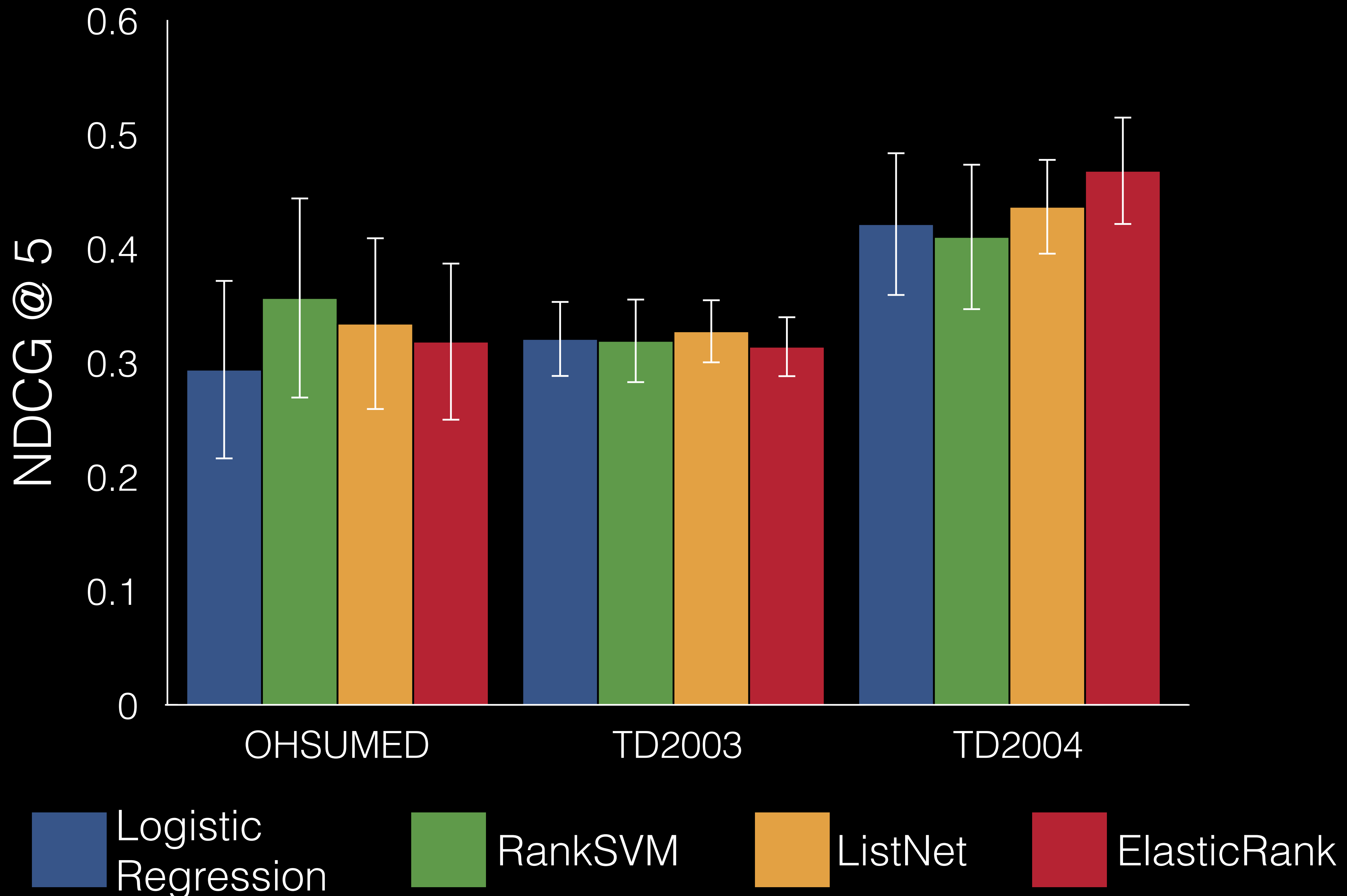
- Prunes every k gradient steps
- If $|w_i| < \theta \Rightarrow w_i = 0$

Hyper-parameter Optimization

- **Turn-key** inference
- Automatic adjustment of hyper-parameters
- Bayesian Approach (Snoek, Larochelle, Adams; 2012)
    - Gaussian Process
    - Thomson Sampling

LETOR Experiments

ElasticRank is comparable with state-of-the-art models



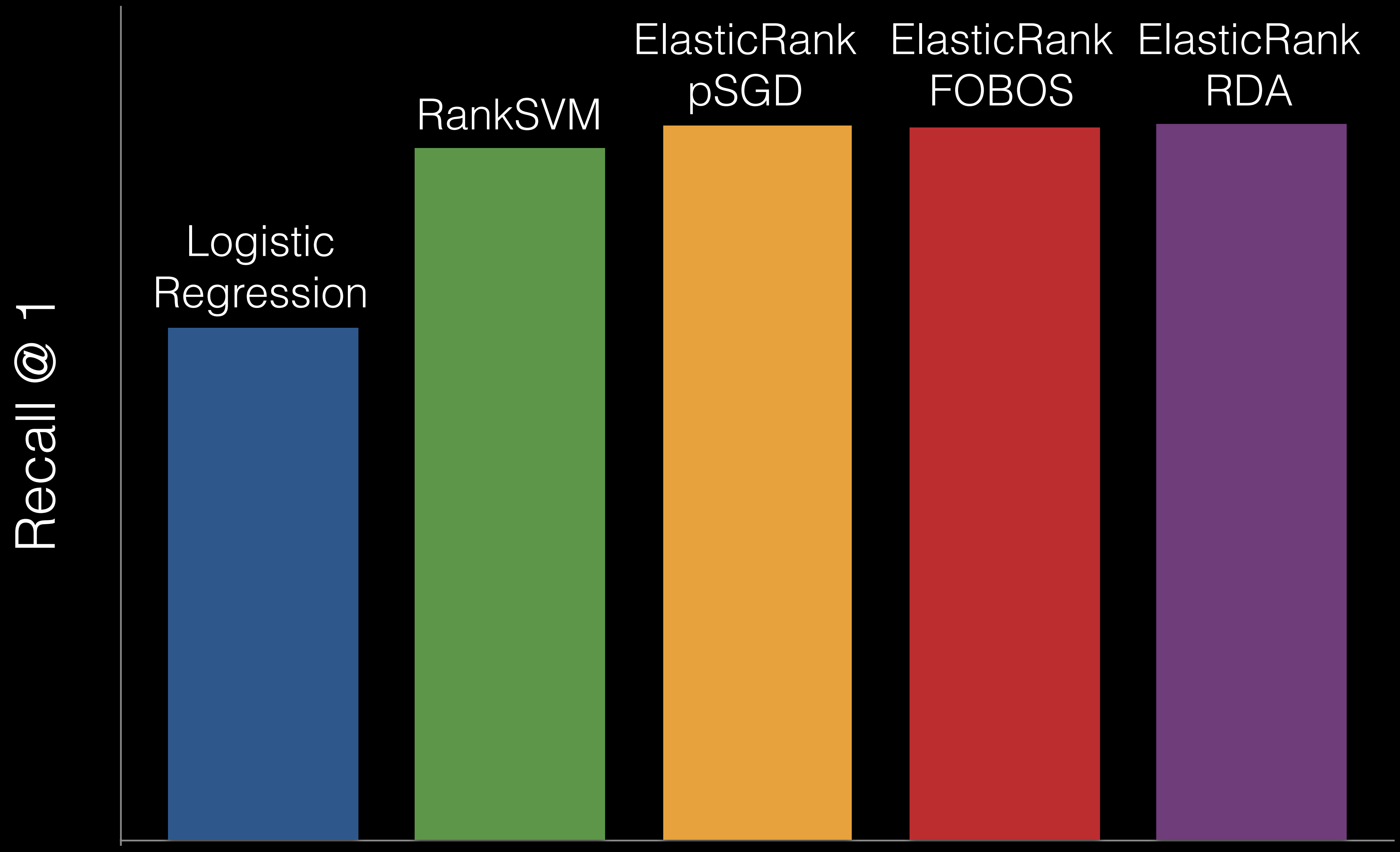
Amazon.com Experiments

Experimental Setup

- # examples ≈ millions
- # features ≈ thousands (millions of dimensions)
- Purchase logs from contiguous time interval

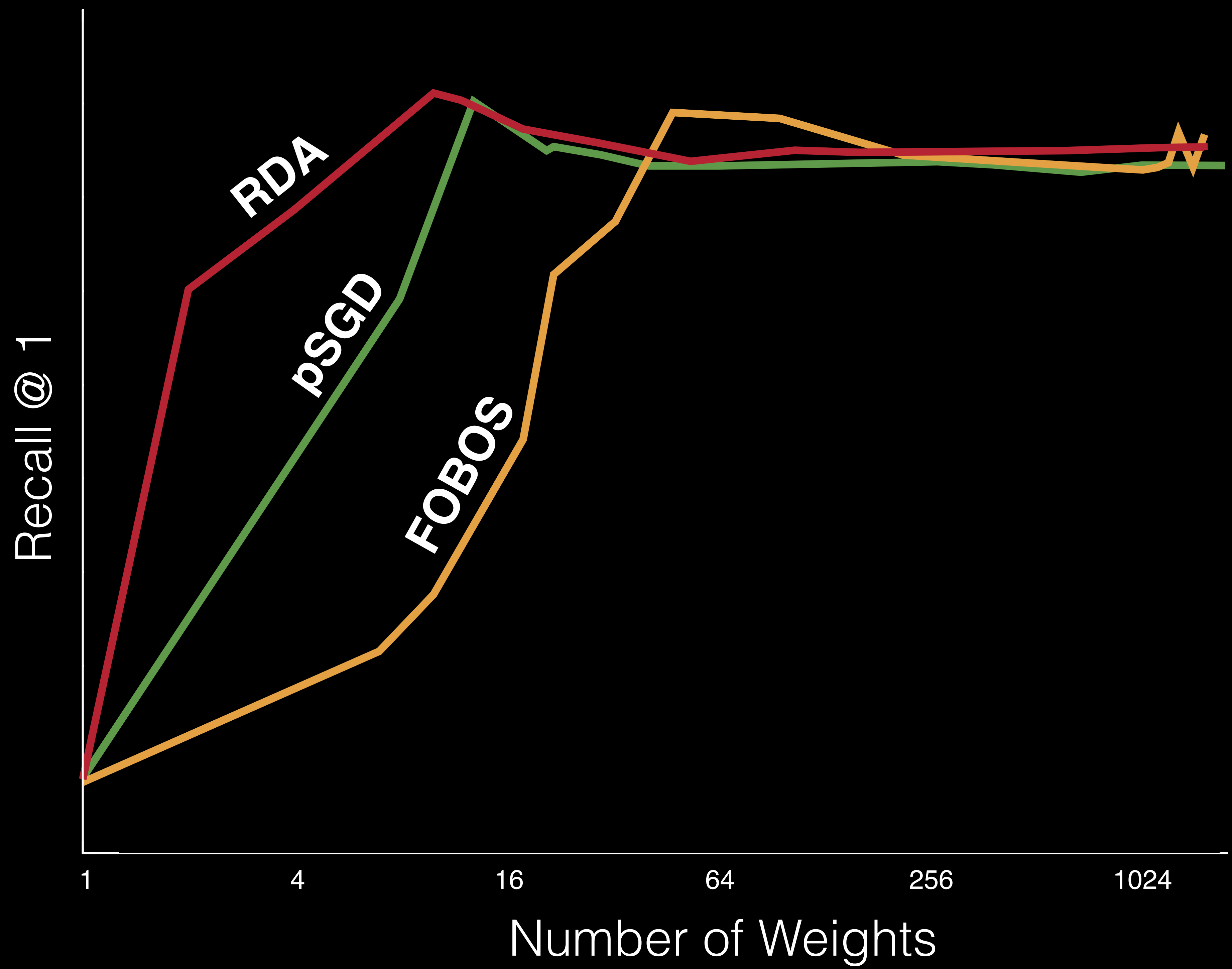
Experimental Results

ElasticRank performs best

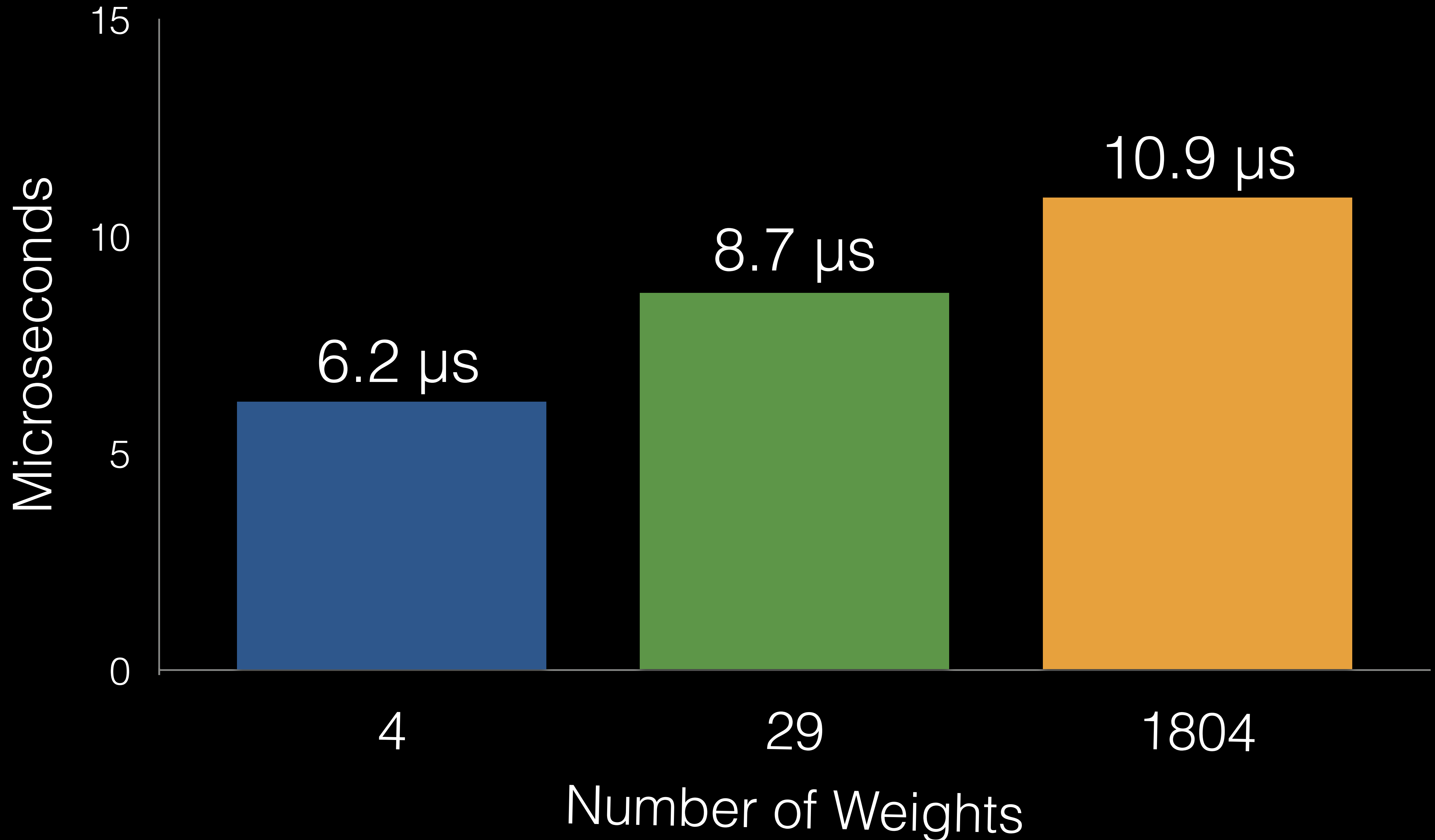


Sparsity vs Performance

RDA achieves the best trade-off



Prediction Time



Contributions

How to learn ranking functions with

- Single pass
- Small memory footprint
- Sparse

WITHOUT sacrificing performance

References

- C. J. C. Burges, R. Ragno, and Q. V. Le. *Learning to rank with nonsmooth cost functions*. In Advances in Neural Information Processing Systems (NIPS), 2006.
- J. C. Duchi and Y. Singer. *Efficient online and batch learning using forward backward splitting*. Journal of Machine Learning Research (JMLR), 2009.
- L. Xiao. *Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization*. Journal of Machine Learning Research (JMLR), 2010.
- J. Snoek, H. Larochelle, and R. P. Adams. *Practical bayesian optimization of machine learning algorithms*. In Advances in Neural Information Processing Systems (NIPS), 2012.

One-Pass Ranking Models for Low-Latency Product Recommendations

Martin Saveski
@msaveski

MIT
(Amazon Berlin)