A Comparative Study on Hyperparameter Optimization for Recommender Systems

Pawel Matuszyk **Renê Tatua Castillo** **Daniel Kottke** **Myra Spiliopoulou**

Knowledge Management & Discovery
Faculty of Computer Science
Otto-von-Guericke University
Magdeburg, Germany

18th of October 2016

Hyperparameter Optimization - Motivation

- ML algorithms often sensitive to hyperparameters
- Hyperparameters are neither trained nor derived from the data
- Setting the right parameters considered a task of a human expert
- Finding reasonable parameter values a priori is difficult (e.g. regularization)
- Goal: finding hyperparameter values that minimize an error metric
- There is no single best optimization method for all problems

Hyperparameter Optimization

Problem Definition

- Let A be the target algorithm
- n number of parameters to be tuned
- Each parameter $\theta_i \in [a_i, b_i], \forall i \in \{1, ..., n\}$
- Parameter configuration space $\Theta = [a_1, b_1] \times \ldots \times [a_n, b_n]$
- Let the vector $\overrightarrow{\theta} = [\theta_1, \theta_2, \ldots, \theta_n]$ represent a parameter configuration
- $H : \Theta \rightarrow \mathbb{R}$ be an error metric that maps $\overrightarrow{\theta}$ to a numeric score

(Notation partially adopted from [Lindawati et al., 2011].)

Optimization Problem

- finding: $\overrightarrow{\theta}^* = \underset{\overrightarrow{\theta} \in \Theta}{\arg\min}\, H(\overrightarrow{\theta})$

Restriction

- Evaluation of H is expensive
- Limited number of samples

pawel.matuszyk@ovgu.de

Fair Comparison of Algorithms

Example

- Let A and B be two algorithms with different parameters
- After the optimization we obtain $\overrightarrow{\theta}^*_A$ and $\overrightarrow{\theta}^*_B$
- Let $H(\overrightarrow{\theta}^*_A) < H(\overrightarrow{\theta}^*_B)$

Case 1 - Manual Tuning by a Human

- Not known, if A is truly better than B, or did the human expert tune A better than B.
- A reliable conclusion is **not** possible.

Fair Comparison of Algorithms II

Case 2 - Semi-automatic Grid Search

- Commonly used in research
- A human expert defines "interesting" parameter values
- An algorithm tests all possible combinations
- More convenient, but the choice of "interesting" parameter values is subjective
- This choice can benefit one algorithm

Case 3 - Fully Automated Optimization

- A human expert only defines intervals $[a_i, b_i]$ and a budget
- Just an interval with working values (not necessarily good values)
- Objective and equal for both algorithms
- Supports replication of results
- **A reliable conclusion possible**

Recommender Systems

- Recommender systems (RS) are a class of ML algorithms
- Designed to ease the problem of information overload
  - many choices for many users (books, friends in social networks, medications, etc.)
  - personalization by learning users' preferences
- Prediction of a rating matrix (or of a tensor)

U\I	i_1	i_2	i_3	i_4	i_5
u_1		4	1		1
u_2	4		3		4
u_3	3		2	5	3
u_4		4		5	

- A commonly used evaluation metric is RMSE
- $H : \Theta \rightarrow \mathit{RMSE}$

pawel.matuszyk@ovgu.de

Hyperparameter Optimization in Recommender Systems

- No study on how to optimize hyperparameters in RS
- Findings of existing studies not transferable
  - SMAC shows a better performance than a random model [Hutter et al., 2011]
  - SMBO significantly better than random search [Bergstra et al., 2011]
  - Genetic algorithm better than random search [Whitley et al., 1998]
  - It does not apply to recommender systems
- The task in RS is different from classification or regression, i.e. a different optimization problem and response surface
- For the evaluation of optimization methods, we use BRISMF [Takács et al., 2009] - Biased Regularized Incremental Simultaneous Matrix Factorization

pawel.matuszyk@ovgu.de

Optimization Algorithms

- Random walk
- Genetic algorithm
- Particle swarm optimization
- Simulated annealing (discrete and Gaussian neighbourhood)
- SMAC [Hutter et al., 2011]
- Random sampling (continuous)
- Random grid sampling (discrete)
- Nelder-Mead
- Full grid

Datasets

- Netflix (2000 users sample)
- ML1M
- ML100k
- Flixter (2000 users sample)

pawel.matuszyk@ovgu.de

Evaluation Settings

- Every optimization algorithm can request 50 settings = 1 optimization run
- Every run is repeated 100 times on a different permutation of a dataset
- Datasets are split into 3 partitions

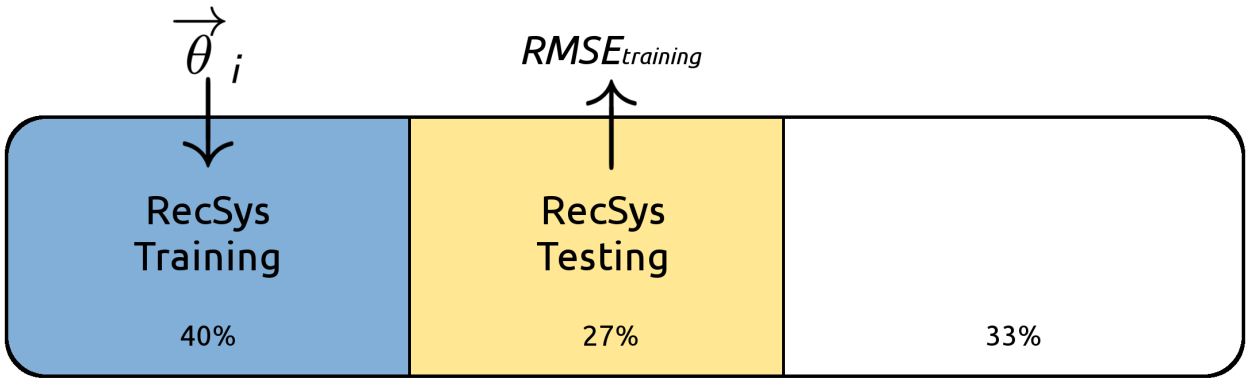


Figure: Dataset splitting for the **training** of hyperparameters.

- Results of 100 runs are aggregated into a median curve
- More than 160,000 experiments

pawel.matuszyk@ovgu.de

Evaluation Settings

- Every optimization algorithm can request 50 settings = 1 optimization run
- Every run is repeated 100 times on a different permutation of a dataset
- Datasets are split into 3 partitions

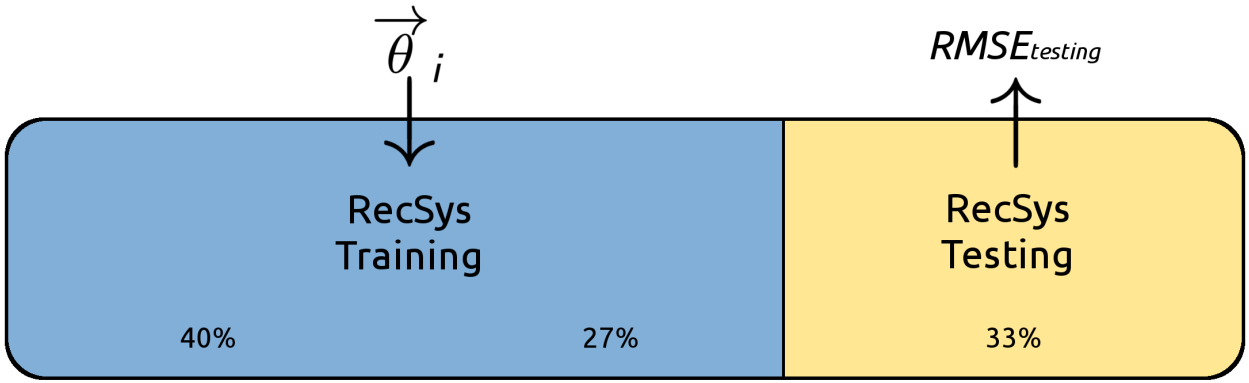


Figure: Dataset splitting for the **testing** of hyperparameters.

- Results of 100 runs are aggregated into a median curve
- More than 160,000 experiments

pawel.matuszyk@ovgu.de

Median Curve

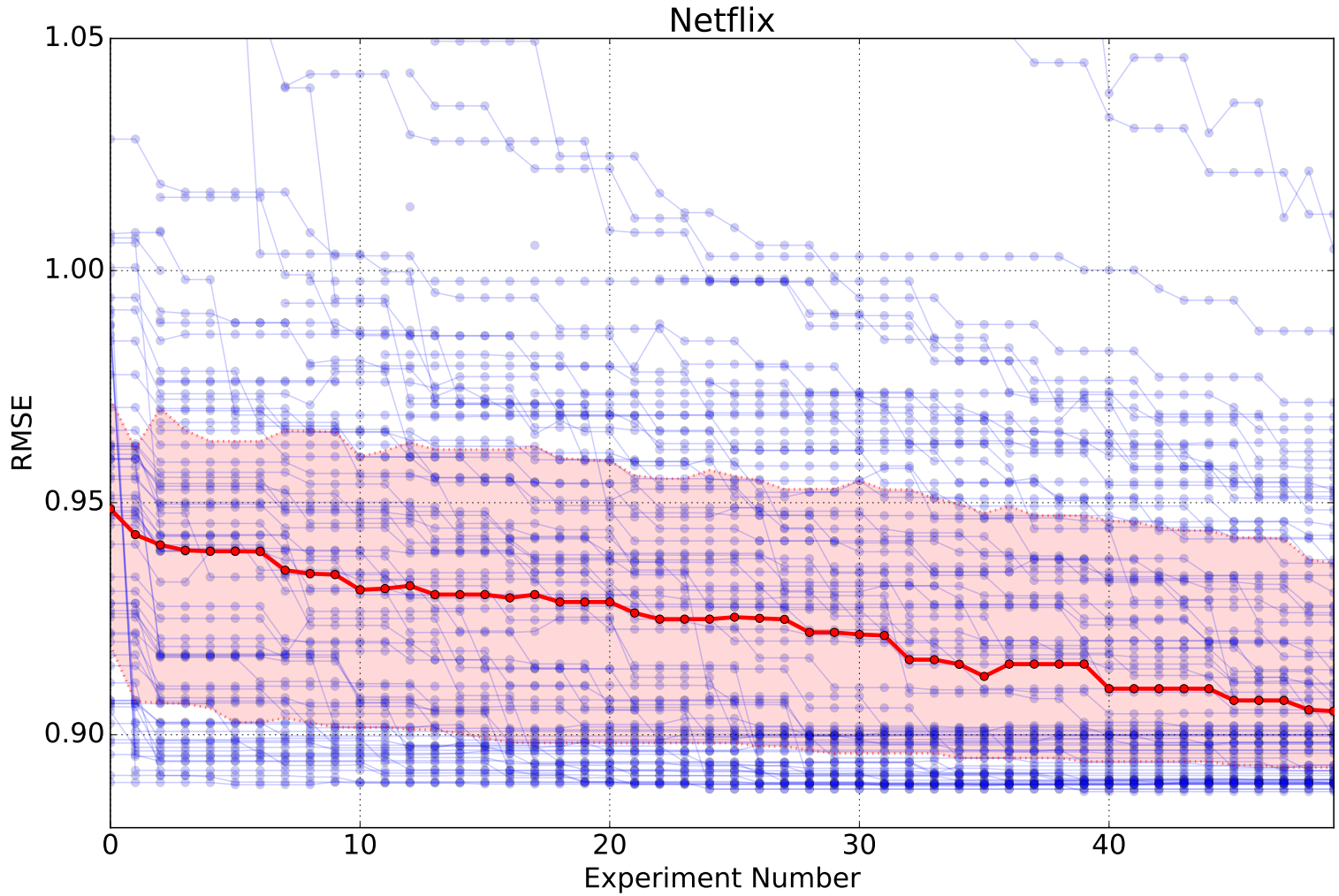


Figure: 100 runs of the random walk algorithm optimizing BRISMF on the Netflix dataset. Shaded area indicates 25th and 75th percentile.

Experimental Results - BRISMF, Netflix

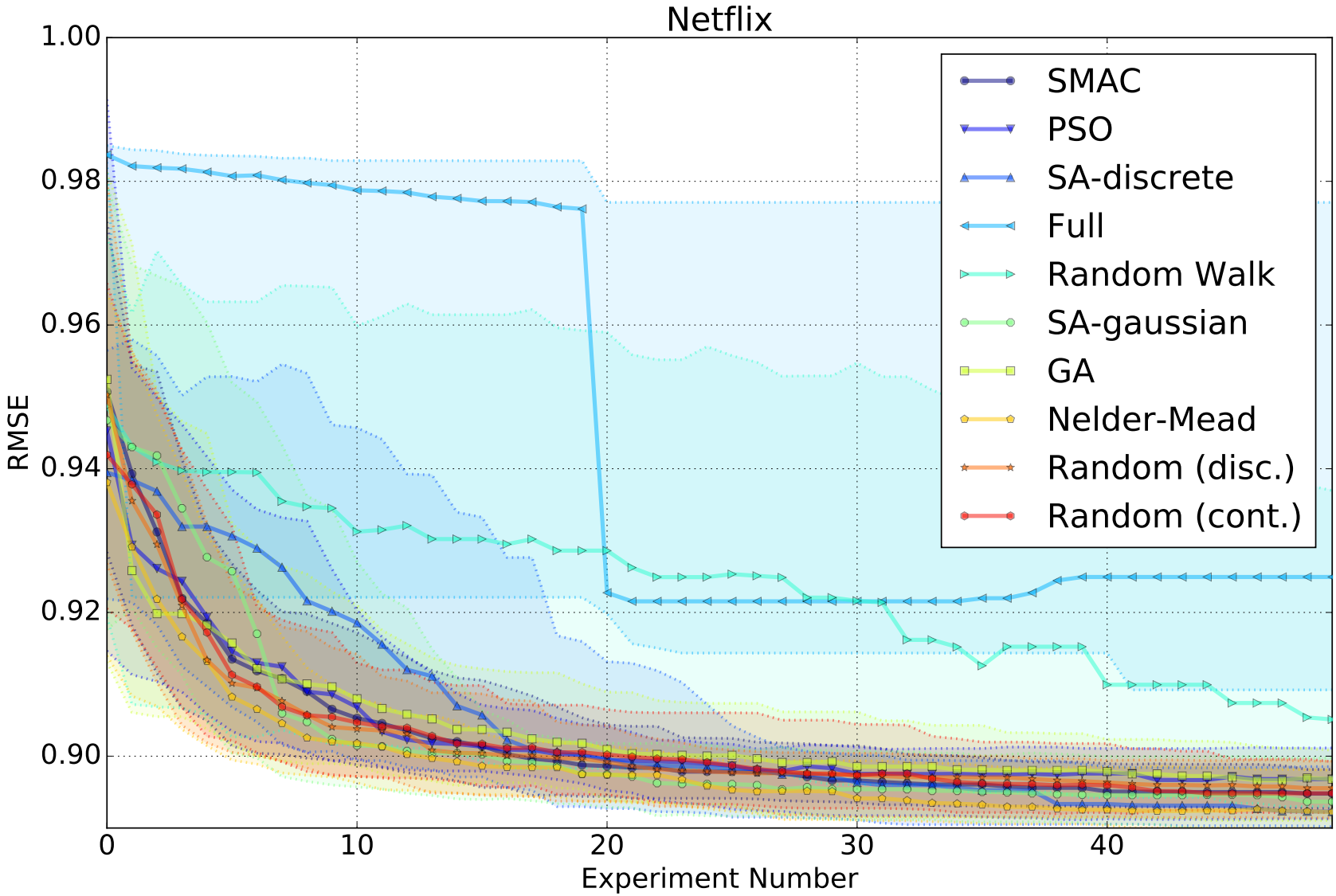


Figure: Optimization of BRISMF on the Netflix dataset.

Experimental Results - BRISMF, ML1M

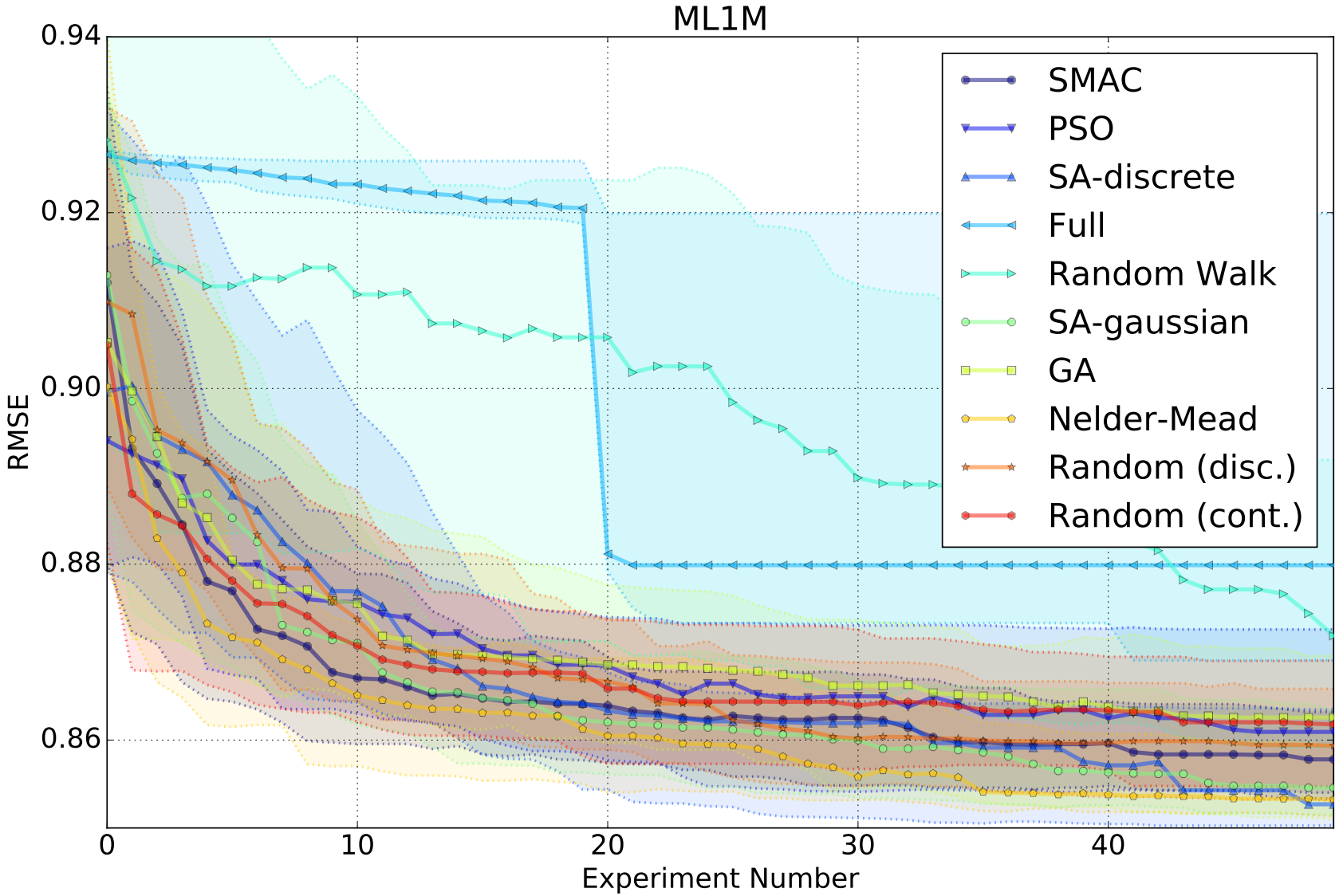


Figure: Optimization of BRISMF on the ML1M dataset.

Experimental Results - BRISMF, Flixter

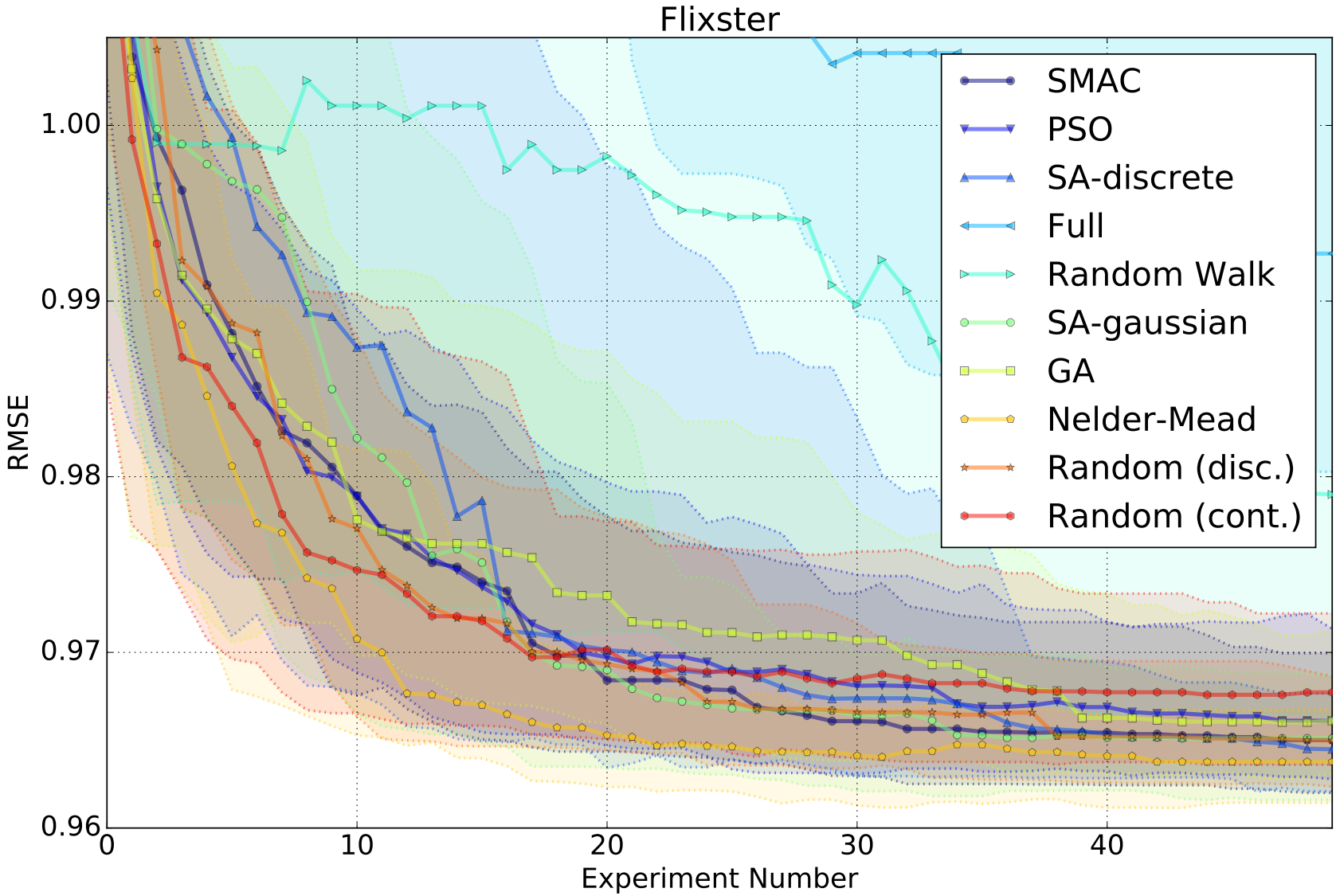


Figure: Optimization of BRISMF on the Flixter dataset.

Experimental Results - BRISMF, ML100k

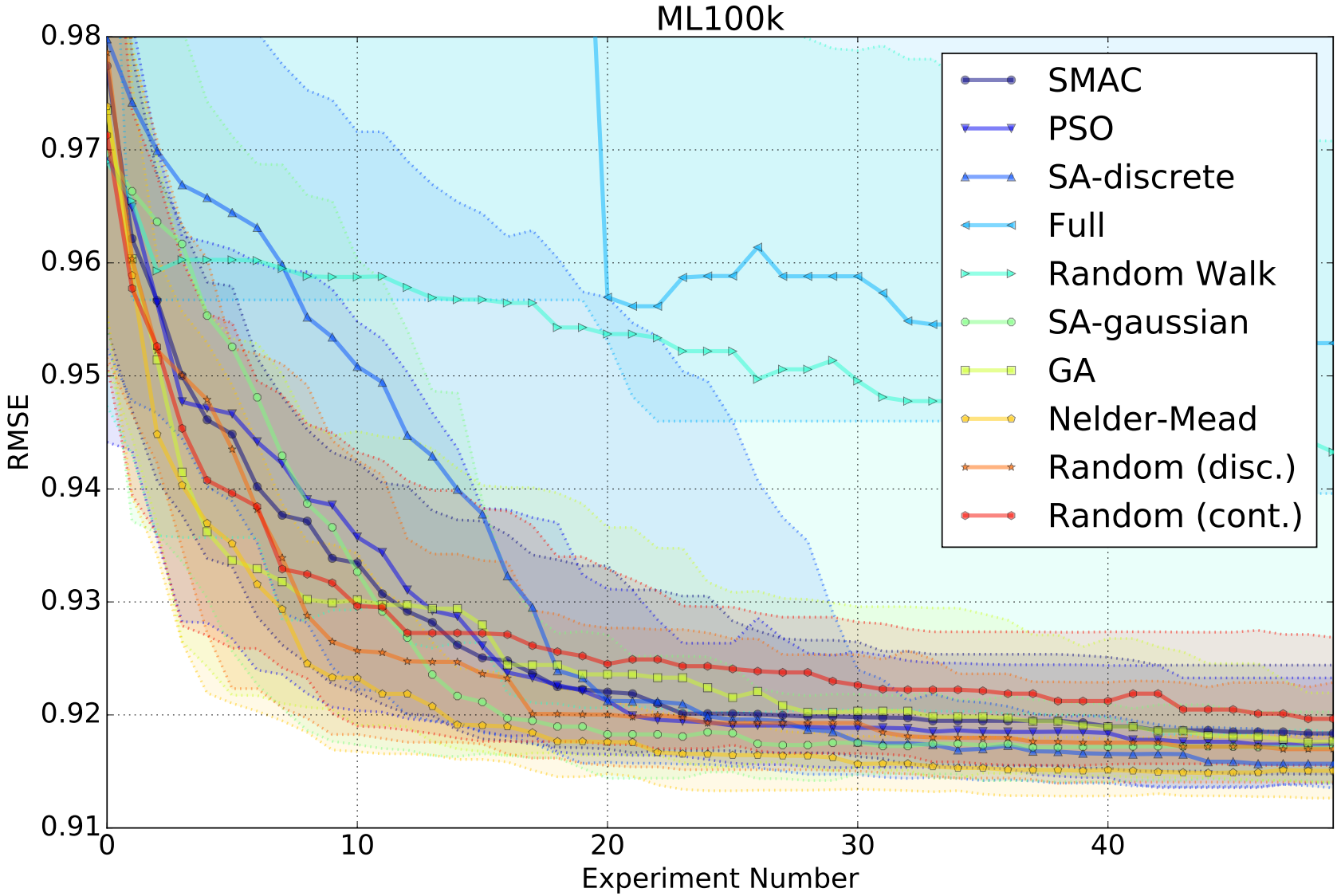


Figure: Optimization of BRISMF on the ML100k dataset.

Interpretation of Results

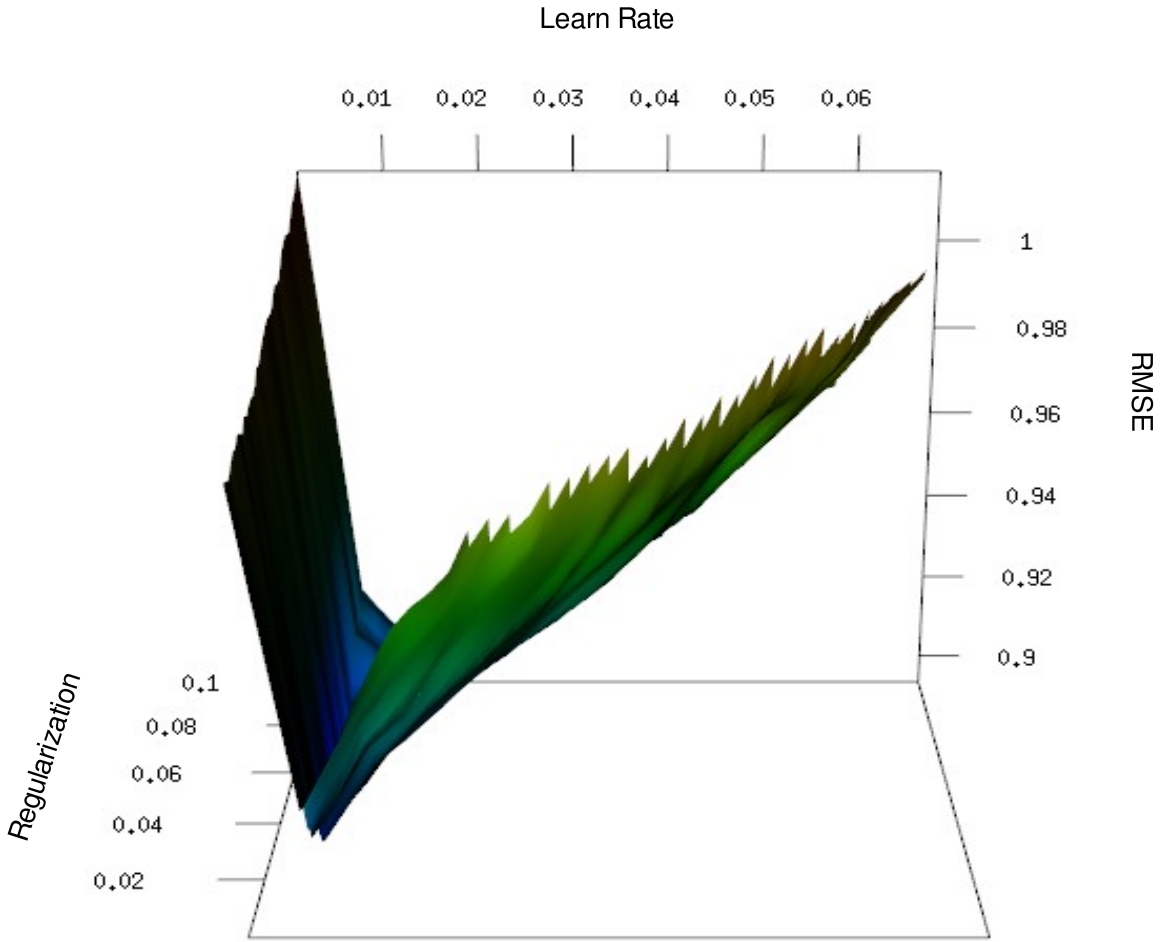
Poorly performing algorithms:

- Random walk
- Exhaustive (full) search - capped after 50 experiments
- SA (discrete) often converged slowly

Well performing algorithms:

- All remaining algorithms performed similarly
- The Nelder-Mead algorithm often marginally better
- Random Selection!

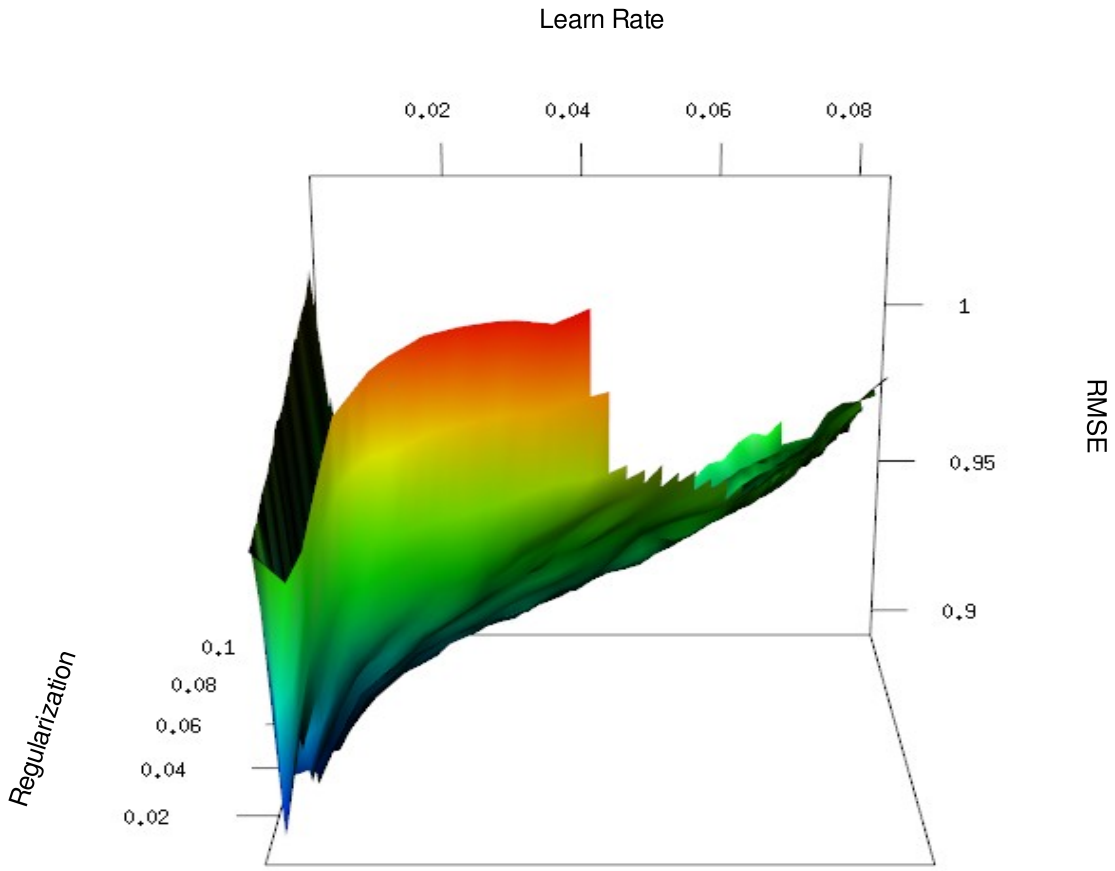
Response Surface



a) Response Surface Dim = 10

Figure: BRISMF algorithm on the Netflix dataset

Response Surface II



e) Response Surface Dim = 150

Figure: BRISMF algorithm on the Netflix dataset

Conclusions

Hyperparameter optimization for RS:

- Does not require sophisticated methods
- Random sampling performs nearly as well as Nelder-Mead and
  - it is fully parallelizable
  - fast
  - easy to implement
  - handles all types of parameters
- Easily parallelizable using commodity hardware (e.g. computer lab) and Hadoop

Bibliography I


Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011).
Algorithms for Hyper-Parameter Optimization.
In *NIPS*.


Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011).
Sequential Model-Based Optimization for General Algorithm Configuration.
In *LION*, LNCS.


Lindawati, Lau, H. C., and Lo, D. (2011).
Instance-Based Parameter Tuning via Search Trajectory Similarity Clustering.
In *LION*.


Salakhutdinov, R. and Mnih, A. (2008).
Probabilistic Matrix Factorization.
In *Advances in Neural Information Processing Systems*, volume 20.


Takács, G., Pilászy, I., Németh, B., and Tikk, D. (2009).
Scalable Collaborative Filtering Approaches for Large Recommender Systems.
J. Mach. Learn. Res., 10.


Whitley, L., Howe, A., Rana, S., Watson, J., and Barbulescu, L. (1998).
Comparing heuristic search methods and genetic algorithms for warehouse scheduling.
In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 3, pages 2430–2435. IEEE.

Thank you!
Do you have questions?