Optimizing Hyper-parameters in Recommender Systems Using Rolling Averages

RODRIGO ALLIENDE, Pontificia Universidad Catolica de Chile, Chile

A clear and well-documented LaTeX document is presented as an article formatted for publication by ACM in a conference proceedings or journal publication. Based on the "acmart" document class, this article presents and explains many of the common variations, as well as many of the formatting elements an author may use in the preparation of the documentation of their work.

ACM Reference Format:
Rodrigo Alliende. 2018. Optimizing Hyper-parameters in Recommender Systems Using Rolling Averages. *J. ACM* 37, 4, Article 111 (August 2018), 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Recommendations systems are widely used in a variety of contexts. Most recommendations systems use hyper-parameters, or parameters that are not changed by the training process. And therefore need to be manually set.

Using the correct hyper-parameters significantly improves the performance of most recommendation systems. Which is why there is a need for methods to find the best hyper-parameters for a given problem.

A very common method for finding parameters is grid search. A method in which for each hyperparameter you assign a range of possible values (usually a geometric or arithmetic progression), and then try all possible combinations of them and comparing their performance on a validation set, a set of data that is different from the set used to train the model. Traditionally the hyper-parameter combination that is used in the final model, is the one that had the best performance in the validation set.

This work seeks to improve the method described above by changing the metric in which the best combination is chosen, by using the rolling averages of the performance instead of the performance.

In the last years investigation on hyper-parameter optimization has focused on updating hyperparameters once new data is obtained [2]

2 PROPOSED METHOD

lets consider a recommender system with hyper-parameters $p^1, p^2 ... p^n$. For each hyper-paramater p^i we will consider a search range $p^i_1, p^i_2 .... p^i_m$. for two search values p^i_j, p^i_k we define the distance between them as follows: $d(p^i_j, p^i_k) = |j - k|$. And given two sets of hyper-parameters we define the distance between them as the maximum pairwise distance.

Next given a set of hyper-parameters in the search range $(p^1_{i1}, p^2_{i2} ..... p^n_{in})$ we define the neighborhood at k distance as all the points $(p^1_{j1}, p^2_{j2} ..... p^n_{jn})$ which have a distance no bigger than k to the original point. And we define the rolling average of any performance metric at that point as the average of that same metric in the neighborhood of the point

Author's address: Rodrigo Alliende, raalliende@uc.cl, Pontificia Universidad Catolica de Chile, Santiago, Chile.

3 DATASETS

The datasets used in this paper are the LastFM user artist dataset and the MovieLens dataset.These are traditional datasets used in recommender systems, these where used because most systems have a reasonable performance in these methods.

4 METHODOLOGY

The models used to test the rolling average hyper-parameter optimization where the following:

BPR: a method that used bayesian analysis of implicit feedback to make recomendations [3]

multiVAE: a deep learning method which uses a variational autoencoder for collaborative filtering using implicit feeback [1]

multiDAE: similar to multiVAE, but uses a denoising encoder [1]

for BPR the hyper-parameters which where changed where number of latent variables, regularization term and learning rate. the search space of each hyper-parameter consisted of 7 different values for a total of 343 total search combinations.

For MultiVAE and MultiDAE the only hyper-parameter which was modified was number of epochs with a total of 100 epochs tested.

For each model and hyper-parameter combination we trained the model on a training set, measured its performance on a validation set and registered the result in a result matrix. Then for each point in the result matrix we calculated the rolling average of its neighborhood at distance 2, and selected the set of hyper-parameters with the highest rolling avereage, and the highest standalone performance. Then both of these models performance was meassured on a holdout (testing) set and the final result was recorded. For BPR the performance metric used was NDCG@20 and for MultiVAE and MultiDAE the performance metric was NDCG@100. NDCG is a common metric to use in recommender systems and the values used where the ones found in the found implementations.

5 RESULTS

Table 1. results for MultiVAE

Dataset	NDCG@100 classical maximum	NDCG@100 rolling avereage maximum
LastFM	0.29602	**0.29981**
Movielens	0.42881	**0.42893**

Table 2. results for MutliDAE

Dataset	NDCG@100 classical maximum	NDCG100 rolling avereage maximum
LastFM	0.34250	**0.34271**
Movielens	0.42077	**0.42104**

Table 3. results for BPR

Dataset	NDCG@20 classical maximum	NDCG@20 rolling avereage maximum
LastFM	0.24518	**0.25963**
Movielens	0.31184	**0.31317**

6 ANALYSIS

In all cases using the rolling average maximum instead of the classical maximum resulted in a performance increase for the given metric. But the performance increase was usually small.The performance increase may be caused by having a model that is less overfitted to the validation set.

The biggest performance increase for deep learning methods were in cases in which the performance didn't stabilize over the number of epochs(MultiVAE on lastFM). In all cases the percentual performance increase was higher in the lastFM dataset which was also the worst performing one.

The number of combinations increases exponentially with the number of hyper-parameters changed, this poses a challenge for models that have many hyper-parameters.

We can see the performance of the model on the validation set with different epoch numbers in the following graphs.

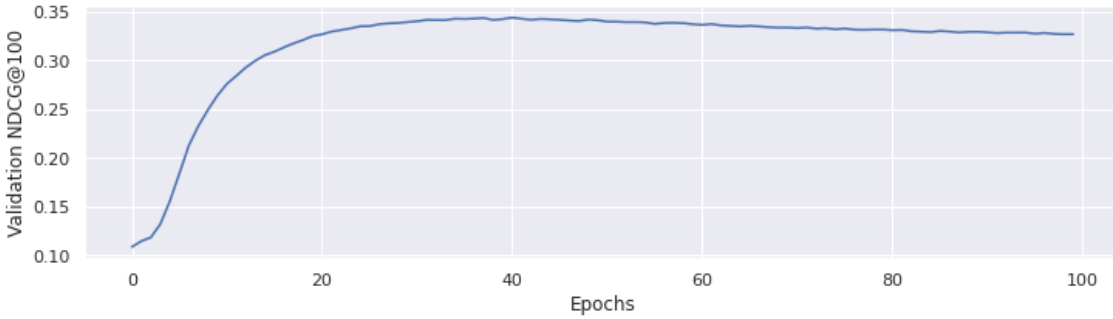

Fig. 1. NDCG vs epoch for multiVAE in last FM

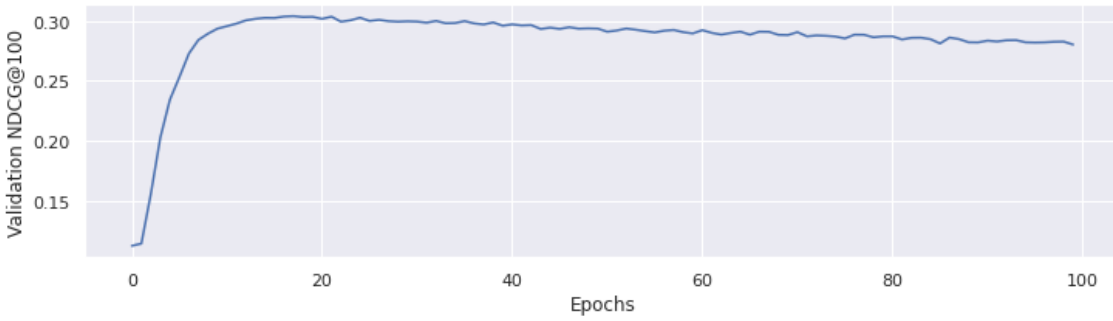

Fig. 2. NDCG vs epoch for multiDAE in last FM

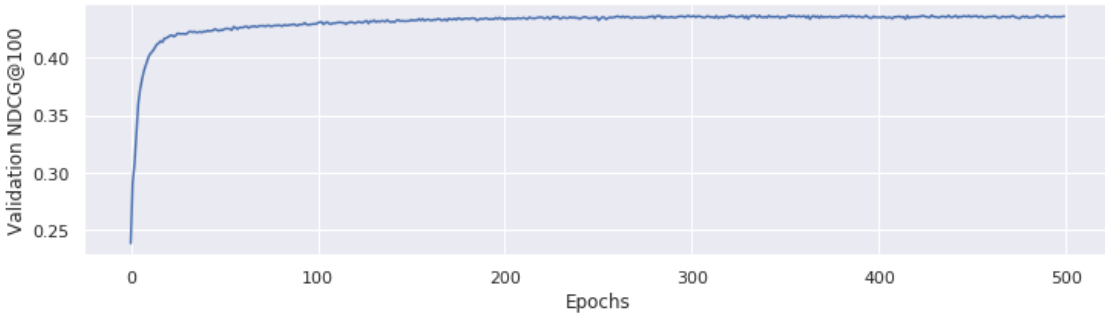

Fig. 3. NDCG vs epoch for multiVAE in movielens

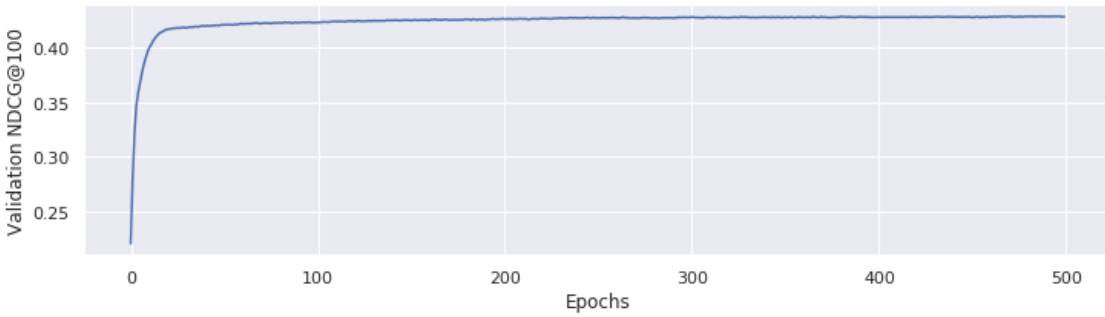

Fig. 4. NDCG vs epoch for multiDAE in movielens

7 CONCLUSIONS

The rolling average method consistently performed slightly better than the traditional method. The method requires to train the same amount of models than the traditional method, resulting in nearly equal time to execute, this is particularly relevant due to the small increase in this method.

The rolling average method is fairly easy to implement. This means that further research on the performance of this optimization method should be easy to do.

While the results were consistently better, the improvement was very small, and further testing is probably needed to be sure that the effect is statistically significant.

Both methods are inappropriate for selecting a large amount of hyper-parameters at the same time due to the exponential increase of combinations in terms of hyper-parameters. for this kind of problems a random sub-sample of the grid is usually tested. And while the rolling average method can be implemented in those cases it has a slower training time.

REFERENCES

[1] Matthew D. Hoffman Tony Jebara Dawen Liang, Rahul G. Krishnan. 2018. *Variational Autoencoders for Collaborative Filtering*. https://arxiv.org/pdf/1802.05814.pdf,lastaccessed=

[2] Licia Capra Simon Chan, Philip Treleaven. 2019. *Continuous Hyperparameter Optimization for Large-scale Recommender Systems*. Retrieved December 13, 2020 from http://www0.cs.ucl.ac.uk/staff/l.capra/publications/bigd-chan.pdf

[3] Zeno Gantner Steffen Rendle, Christoph Freudenthaler and Lars Schmidt-Thieme. 2009. *BPR: Bayesian Personalized Ranking from Implicit Feedback*. https://arxiv.org/ftp/arxiv/papers/1205/1205.2618.pdf,lastaccessed=