Self Hyper-parameter Tuning for Stream Recommendation Algorithms

Bruno Veloso[1,2], João Gama[1,3], Benedita Malheiro[4,5], and João Vinagre[1,6]

[1] LIAAD - INESC TEC, Porto, Portugal
[2] UPT - University Portucalense, Porto, Portugal
[3] FEP - University of Porto, Porto, Portugal
[4] ISEP - Polytechnic of Porto, Porto, Portugal
[5] CRAS - INESC TEC, Porto, Portugal
[6] FCUP - University of Porto, Portugal

Abstract. E-commerce platforms explore the interaction between users and digital content – user generated streams of events – to build and maintain dynamic user preference models which are used to make meaningful recommendations. However, the accuracy of these incremental models is critically affected by the choice of hyper-parameters. So far, the incremental recommendation algorithms used to process data streams rely on human expertise for hyper-parameter tuning. In this paper we apply our Self Hyper-Parameter Tuning (SPT) algorithm to incremental recommendation algorithms. SPT uses the Nelder & Mead optimisation algorithm to perform hyper-parameter tuning. It creates three models with different hyper-parameters, assesses them at dynamic size intervals and applies the Nelder & Mead operators to update their hyper-parameters until they converge. The main contribution of this work is the adaptation of the SPT method to incremental matrix factorisation recommendation algorithms. The proposed method was evaluated with well-known recommendation data sets. The results show that SPT systematically improves data stream recommendations.

Keywords: Parameter Tuning, Hyper-parameters, Optimisation, Nelder-Mead, Recommendation

1 Introduction

With the increase of strategic information retained by businesses, the adoption of machine learning algorithms is essential to retrieve valuable information and increase profits. However, these machine learning tools still face a set of complex problems such as on-line hyper-parameter optimisation and model selection.

The hyper-parameter optimisation problem has been addressed in the literature using grid-search [12], random-search [1] and gradient descent [19] algorithms. So far, these approaches have been applied to off-line scenarios since they require train and validation stages. To overcome this limitation, this work focus on the on-line hyper-parameter optimisation for stream-based recommendation.

Self Parameter Tuning (SPT) is a direct-search hyper-parameter optimisation algorithm based on the Nelder-Mead algorithm [23] and dynamic data stream samples. Our proposal applies SPT [28] to stream-based recommendation, continuously searching for the optimal learning rate and regularisation parameter, *i.e.*, for the best incremental matrix factorisation model.

The contribution of this paper is the application of the SPT algorithm to incremental recommendation algorithms. Our extension of the Nelder-Mead algorithm not only processes successfully recommendation problems, but is, to the best of our knowledge, the single one which effectively works with data streams.

The rest of the paper is organised as follows: Section 2 describes the related automatic machine learning work; Section 3 presents the proposed solution for the identified problem; Section 4 describes the experiments and discusses the results obtained; and Section 5 presents the conclusions and suggests future developments.

2 Related Work

In machine learning, the ability to select appropriate features, work flows, machine learning paradigms, algorithms, and their hyper-parameters requires expert knowledge [13]. The few contributions found in the literature addressing this progressive automation of machine learning or auto-ML include tools [1,27,9], model selection algorithms [7,6], hyper-parameter optimisation algorithms [18,10,24] and Nelder-Mead optimisation solutions [16,8,25].

These on-line auto-ML tools adopt Bayesian optimisers to tune the hyper-parameters of the specified model [1,27,9]; [1,27] use cross-validation to guide the search direction; and [9] takes into account the performance on similar data sets to improve the efficiency of the algorithm. In terms of automatic hyper-parameter selection, there are several different techniques: (*i*) particle swarm optimisation [7], which is flexible and can be applied to ensemble models [6]; (*ii*) grid search [18], which minimises the estimated error until converges on a local minima; (*iii*) gradient-based search, *e.g.*, Stochastic Gradient Descent (SGD), which converges to an optimal solution [24]; and (*iv*) Nelder-Mead direct search, which relies on heuristics to optimise model parameters [16] or tensor based models [8]. The Nelder-Mead algorithm has been used together with exponentially decay centrifugal forces to improve the results at the cost of the number of iterations needed to converge [15] as well as with reinforcement techniques (e-greedy) to select the best model of each iteration [25].

Our proposal differs from all the above because it operates on-line by automatically adjusting the hyper-parameters of the models based on the stream of events. Nevertheless, it is also applicable to off-line batch learning.

3 Self Parameter Tuning

This paper presents the application of the SPT algorithm to stream-based recommendation algorithms. The SPT algorithm was designed to optimise a set

of hyper-parameters,namely the learning rate and regularisation parameter. We adopt a direct-search algorithm, that uses heuristics to avoid algorithms which rely on hyper-parameters. Specifically, we adapt the Nelder-Mead method [23] to work with stream-based recommendation algorithms. Figure 1 represents the

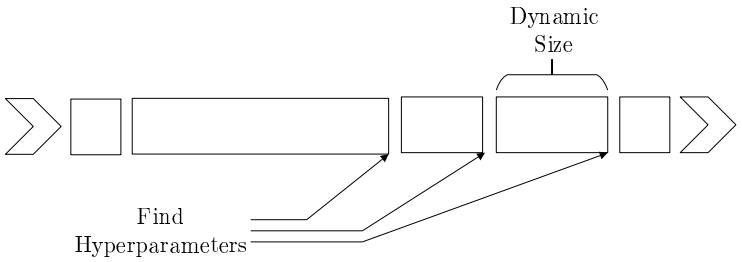


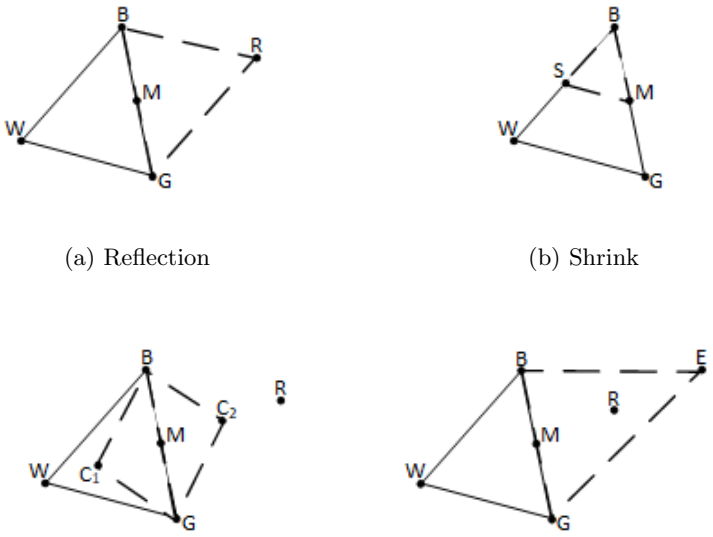
Fig. 1. Application of the proposed algorithm to the data stream.

application of the proposed algorithm. In particular, to find a solution for n hyper-parameters, it requires $n + 1$ input models, *e.g.*, to optimise two hyper-parameters, the algorithm needs three alternative input models, corresponding to the three vertexes of the Nelder-Mead algorithm. The models are initialised with randomly selected learning rate and regularisation parameters, and the Nelder-Mead operators are then applied over dynamic sample intervals. The algorithm processes each data stream sample, using the three models until they converge. The following subsections describe the implemented Nelder-Mead algorithm, including the dynamic sample size selection.

3.1 Nelder-Mead Optimization

This algorithm is a simplex search algorithm for multidimensional unconstrained optimization without derivatives. The vertexes of the simplex, which define a convex hull shape, are iteratively updated in order to sequentially discard the vertex associated with the largest cost function value.

The Nelder-Mead algorithm applies four simple operations to the three vertexes (models): *reflection*, *shrinkage*, *expansion* and *contraction* (Figure 2). The three vertexes are ordered by root mean square error (RMSE) value: best (B), good (G), which is the second best, and worst (W). While Algorithm 1 implements the reflection and extension operations, Algorithm 2 addresses the contraction and shrinkage operations. Each operation computes an additional set of vertexes (midpoint M, reflection R, expansion E, contraction C and shrinkage S) and ensures they belong to the search space. First, Algorithm 1 determines the midpoint (M) of the best side of the triangle – connecting the best vertex (B) and the good vertex G – as well as the reflection point (R). After this initial step, it heuristically decides whether to reflect or expand (lines 3, 4 and 8).



(a) Reflection (b) Shrink

(c) Contraction (d) Expansion

Fig. 2. Nelder-Mead Operations

Algorithm 1 Nelder-Mead - reflect or expand

1: $M = (B + G)/2$
2: $R = 2M - W$
3: **if** $f(R) < f(G)$ **then**
4: **if** $f(B) < f(R)$ **then**
5: $W = R$
6: **else**
7: $E = 2R - M$
8: **if** $f(E) < f(B)$ **then**
9: $W = E$
10: **else**
11: $W = R$
12: **end if**
13: **end if**
14: **end if**

Algorithm 2 calculates the contraction point (C) of the worst side of the triangle – the midpoint between the worst vertex (W) and the midpoint M – and shrinkage point (S) – the midpoint between the best (B) and the worst (W) vertexes. Then, it determines whether to contract or shrink based on the set of predetermined heuristics (lines 3, 4, 8, 12 and 15).

In this case, we intend to optimise the learning rate and the regularisation parameter, which are constrained to values between 0 and 1. The violation of this constraint results in the adoption of the nearest lower or upper bound.

Algorithm 2 Nelder-Mead - contract or shrink

```
1: M = (B + G)/2
2: R = 2M − W
3: if f(R) ≥ f(G) then
4:     if f(R) < f(W) then
5:         W = R
6:     else
7:         C = (W + M)/2
8:         if f(C) < f(W) then
9:             W = C
10:        else
11:            S = (B + W)/2
12:            if f(S) < f(W) then
13:                W = S
14:            end if
15:            if f(M) < f(G) then
16:                G = M
17:            end if
18:        end if
19:    end if
20: end if
```

3.2 Adaptive Sample Size

The outcome of the Nelder-Mead algorithm depends on the sample size. We calculate a dynamic sample size based on the RMSE metric every time the Nelder-Mead tries to find an optimal solution. The sample size S_{size} is given by Equation 1 where σ represents the RMSE standard deviation and M is the desired error margin. We use 95 % in our experimental work.

$$S_{size} = \frac{4\sigma^2}{M^2} \tag{1}$$

However, to avoid using small samples, we defined a lower bound of 30 samples.

4 Experimental Evaluation

The following subsections describe the experiments performed, including the data sets, the evaluation metrics and protocol, the tests and the results. The experiments were performed with an Intel Xeon CPU E5-2680 2.40 GHz Central Processing Unit (CPU), 32 GiB DDR3 Random Access Memory (RAM) and 1 TiB of hard drive platform running the Ubuntu 16.04. The SPT approach was compared against a default hyper-parameter initialisation – hereafter called baseline. The baseline hyper-parameter initialisation was, 1.0 for the learning rate and 0.05 for the regularisation parameters.

4.1 Data Sets

For the experiments, we selected the following recommendation data sets: (*i*) MovieLens 100k (ML100k) [21] contains information about 943 users and 1682 movies, including 100 000 user ratings together with timestamps; (*ii*) MovieLens

1M (ML1M) [22] holds information about 6040 users and 3900 movies, including 1 000 209 user ratings together with timestamps; (*iii*) Jester [2] data set stores information about 59 132 users and 150 jokes, including 1.7 million user ratings; and (*iv*) GoodBooks [14] data set contains information on 10 000 books, including 1 000 000 user ratings.

4.2 Evaluation Metrics and Protocol

The evaluation protocol defines the data ordering, partitioning, distribution and evaluation metrics. To evaluate the proposed method we applied two different protocols: holdout evaluation [17] and the predictive sequential (prequential) evaluation [11]. The holdout evaluation protocol is used to find an optimal solution for the hyper-parameters and verify the reproducibility of the algorithm. Then, we apply the prequential evaluation to the data as a stream to assess the performance of our method.

In terms of evaluation metrics we adopt the incremental RMSE adopted by Takács *et al.* (2009) [26], which is calculated incrementally after each new viewer rating event. Additionally, we calculate incrementally the Recall@N proposed by Cremonesi *et al.* [4]. For each new event, we randomly select 1000 items not yet rated by the active user, add the newly rated item and, then, make predictions for this subset of 1001 items. Finally, we sort these 1001 items by descending prediction value and, if the newly rated item belongs to the list of the top N viewer predicted items, we count a hit.

Figure 3 presents holdout data partition. The data is ordered temporally and, then, partitioned in two halves: 50 % to "Train" and the remaining 50 % to "Test". First, the holdout algorithm finds an optimal solution for the selected hyper-parameters using the train data. Then, it builds a model using the train data and the identified optimal hyper-parameters. Finally, the holdout algorithm updates and evaluates the created model using the test data. The holdout protocol was repeated 30 times to compute the average and standard deviation of the evaluation metrics.

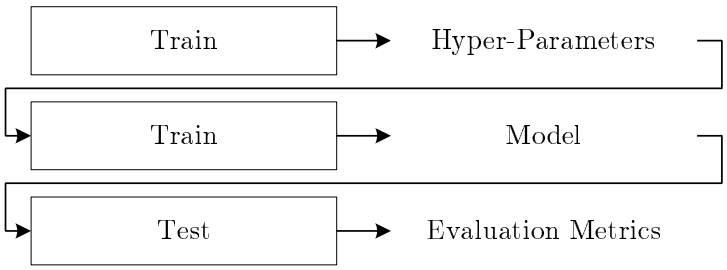


Fig. 3. Holdout – data splitting and processing.

In the case of the prequential protocol, the entire data is simultaneously used for training and testing as represented in Figure 4. First, the data are ordered temporally, then, they are used to build incrementally the three models and, finally, the results are evaluated with a sliding window of 1000 instances. In order to produce the best recommendations, the prediction model used throughout the experiment is dynamic. In fact, it corresponds to the best model found so far by the SPT algorithm.

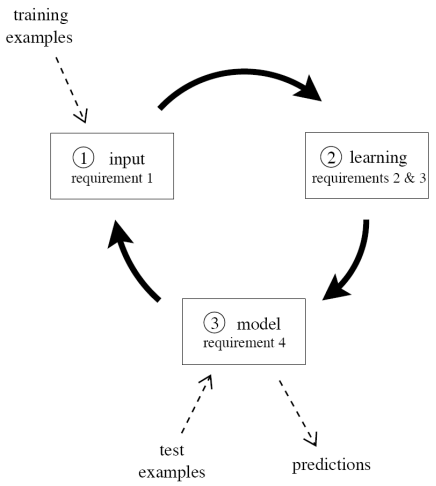


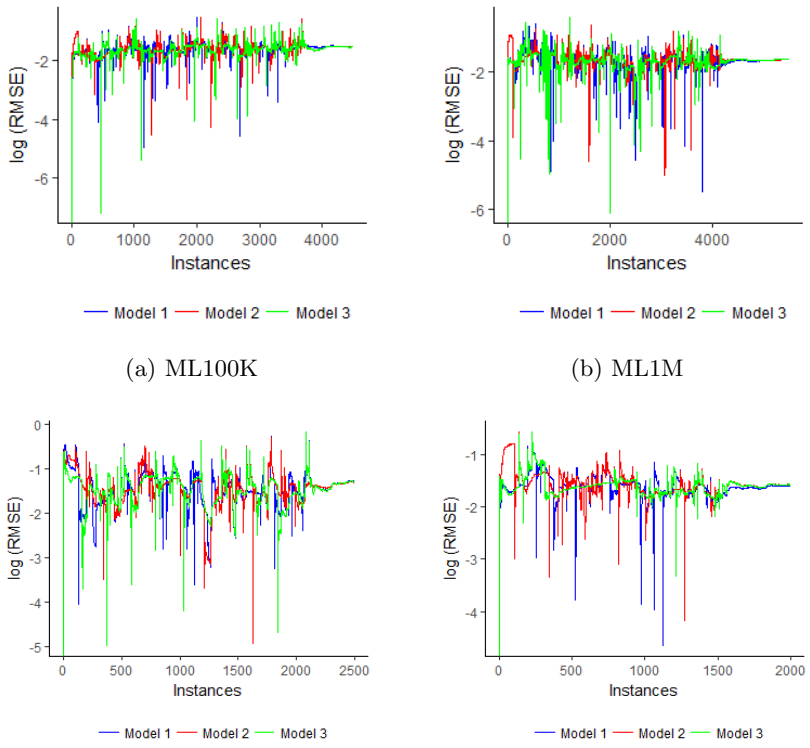
Fig. 4. Prequential – data splitting and processing [3].

4.3 Significance Tests

To detect the statistical differences between the proposed and the baseline approaches we applied three different significance tests: (*i*) the Wilcoxon test [29] to verify if the mean ranks of two samples differ; (*ii*) the McNemar test [20] to assess if a statistically significant change occurs on a dichotomous trait at two time points on the same population; and (*iii*) the critical distance measure proposed by [5] for a graphical interpretation of the statistical results. We define a 5 % of significance level for all tests. The goal of the Wilcoxon and McNemar tests is to reject the null-hypothesis, *i.e.*, that both approaches have the same performance. We run 30 trials for each experiment. At 5 % significance, the critical value of McNemar test (MT_{crit}) is 3.84 and the critical value of the Wilcoxon test (WT_{crit}) is 137. In the case of the McNemar, two samples are statistically different if $MT_{stat} > MT_{crit}$, whereas, in the case of Wilcoxon, two samples are statistically different if the $|WT_{stat}| > WT_{crit}$.

4.4 Experiments

The goal is to optimise the learning rate and regularisation hyper-parameters of the recommendation algorithm proposed by Takács *et al.* (2009) [26]. First, we created three identical initial models with randomly selected learning rate and regularisation values and, then, applied our hyper-parameter optimisation algorithm. Figure 5 shows that the convergence of the three recommendation models occurs in less than 5000 events with all data sets.



(a) ML100K (b) ML1M

(c) Jester (d) GoodBooks

Fig. 5. Recommendation - Model convergence

After the verification of the model convergence, we applied the holdout evaluation protocol to assess the performance of the algorithm with the new hyper-parameters. The experiment was performed 30 times to compute the average and standard deviation of the RMSE and Recall@10 for the SPT and baseline (B) approaches. Table 1 not only displays these results, but highlights for each data set the best case of each evaluation metric, including the corresponding coefficients of variation (CV). The ML100k data set displays a RMSE decrease of 1.4 %

and a Recall@10 increase of 2.1 %. With ML1M, the prediction error decreases 1.9 % and the Recall@10 drops 1.9 %. The Jester data set shows an improvement of 5.5 % and 6.6 % in terms of RMSE and Recall@10, respectively. Finally, the GoodBooks data set presents a decrease of 2.8 % in RMSE and an increase of 1.0 % in Recall@10. Regarding the holdout results, the statistical results for the

Table 1. Recommendation – Holdout results

Dataset	Approach	Metric	μ	CV (%)
ML100K	B	RMSE	2.046×10^{-1}	0.074
		Recall@10	0.097×10^{-1}	3.166
	SPT	RMSE	2.018×10^{-1}	0.066
		Recall@10	0.099×10^{-1}	3.427
ML1M	B	RMSE	2.016×10^{-1}	0.048
		Recall@10	0.106×10^{-1}	1.908
	SPT	RMSE	1.978×10^{-1}	0.038
		Recall@10	0.104×10^{-1}	1.672
Jester	B	RMSE	2.400×10^{-1}	0.226
		Recall@10	0.855×10^{-1}	0.265
	SPT	RMSE	2.269×10^{-1}	0.192
		Recall@10	0.911×10^{-1}	0.284
GoodBooks	B	RMSE	1.952×10^{-1}	0.013
		Recall@10	0.988×10^{-1}	0.325
	SPT	RMSE	1.897×10^{-1}	0.023
		Recall@10	0.996×10^{-1}	0.449

Wilcoxon and McNemar tests reject the null hypothesis, regardless of the data set. The McNemar test statistic value (MT_{stat}) is 28.03 which corresponds to a p-value of 1.19×10^{-7} and the Wilcoxon statistic value (WT_{stat}) is 465 with a p-value of 1.86×10^{-9}. The calculated McNemar and Wilcoxon test statistic values are higher than the corresponding reference values for p-value=0.05 and their significance levels are smaller than 0.05. Figure 6 plots the critical distance between the proposed and baseline optimisation algorithms, showing that they are statistically different. The critical distance between both approaches was determined using the Nemenyi test.

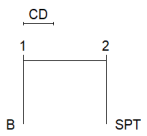
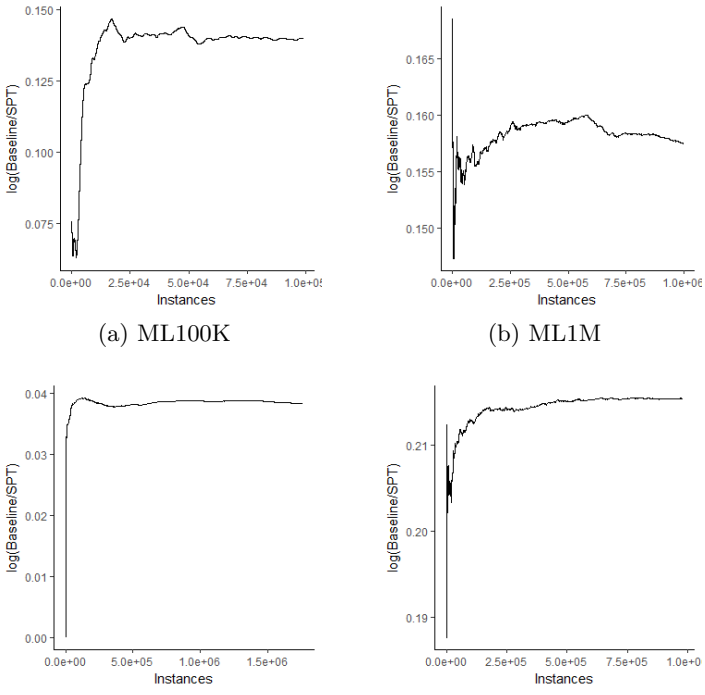


Fig. 6. Recommendation – Holdout Critical Distance

The prequential evaluation shows that the proposed dynamic model outperforms the baseline approach. Figure 7 displays the relative RMSE results between the baseline and SPT methods. Considering the Recall@10, there is an increase of 1.8 % with ML100k, 16.5 % with ML1M, 6.3 % with Jester and a decrease of 19.5 % in the case of GoodBooks.



(a) ML100K (b) ML1M

(c) Jester (d) GoodBooks

Fig. 7. Recommendation – Relative RMSE prequential results

5 Conclusions

This paper describes the application of SPT to incremental recommendation algorithms. In this case, SPT was used to find dynamically the best learning rate and regularisation hyper-parameters.

The main contribution of this paper is an extension of the Nelder-Mead optimisation algorithm to stream-based recommendation. The SPT algorithm is, in terms of existing hyper-parameter optimisation algorithms, less computationally expensive than Bayesian optimisers, stochastic gradients or even grid search algorithms. This proposal is, to the best of our knowledge, the single one which effectively works with data streams in a recommendation scenario.

Taking into consideration that the selection of the hyper-parameters has a substantial impact on the outcome of recommendation algorithms, we applied the proposed method and studied its performance in with holdout and prequential evaluation protocols. The results shows that, not only our algorithm converged rapidly with both evaluation protocols, but also outperformed the baseline results.

Future work will includes three key points: (*i*) application of the algorithm to classification algorithms; (*ii*) selection of machine learning models; and (*iii*) thorough comparison with other optimisation algorithms.

Acknowledgements

This research was carried out in the framework of the project TEC4Growth – RL SMILES – Smart, mobile, Intelligent and Large Scale Sensing and analytics NORTE-01-0145-FEDER-000020 which is financed by the north Portugal regional operational program (NORTE 2020), under the Portugal 2020 partnership agreement, and through the European regional development fund.

References

1. J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(1):281–305, Feb. 2012.
2. Berkeley University. Jester data set. Accessed on March 2018.
3. A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.
4. P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.
5. J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, Dec. 2006.
6. H. J. Escalante, M. Montes, and E. Sucar. Ensemble particle swarm model selection. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.
7. H. J. Escalante, M. Montes, and L. E. Sucar. Particle swarm model selection. *Journal of Machine Learning Research*, 10(Feb):405–440, 2009.
8. S. Fernandes, H. F. Tork, and J. Gama. The initialization and parameter setting problem in tensor decomposition-based link prediction. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 99–108, Oct 2017.
9. M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pages 2962–2970, 2015.
10. C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

11. J. Gama, R. Sebastião, and P. P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, Mar. 2013.
12. C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al. A practical guide to support vector classification, 2003.
13. F. Hutter, R. Caruana, R. Bardenet, M. Bilenko, I. Guyon, B. Kégl, and H. Larochelle. AutoML Workshop @ ICML'14, 2014. Accessed on 18 July 2018.
14. Kaggle. Goodbooks data set. Accessed on March 2018.
15. R. Kar, A. Konar, A. Chakraborty, A. L. Ralescu, and A. K. Nagar. Extending the nelder-mead algorithm for feature selection from brain networks. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 4528–4534. IEEE, 2016.
16. N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 165–172. ACM, 2011.
17. R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
18. R. Kohavi and G. H. John. Automatic parameter selection by minimizing estimated error. In *Machine Learning Proceedings 1995*, pages 304–312. Elsevier, 1995.
19. D. Maclaurin, D. Duvenaud, and R. P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2113–2122. JMLR.org, 2015.
20. Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, Jun 1947.
21. MovieLens. Movielens 100k data set. Accessed on March 2018.
22. MovieLens. Movielens 1M data set. Accessed on March 2018.
23. J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.
24. A. Nichol and J. Schulman. Reptile: a Scalable Metalearning Algorithm. *ArXiv e-prints*, Mar. 2018.
25. P. Pfaffe, M. Tillmann, S. Walter, and W. F. Tichy. Online-autotuning in the presence of algorithmic choice. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1379–1388. IEEE, 2017.
26. G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, 10:623–656, June 2009.
27. C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 847–855, New York, NY, USA, 2013. ACM.
28. B. Veloso, B. Malheiro, and J. Gama. Self hyper-parameter tuning for data streams. In L. Soldatova, J. Vanschoren, M. Ceci, and G. A. Papadopoulos, editors, *Discovery Science*, volume 11198 of *Lecture Notes in Artificial Intelligence*, pages 241–255. Springer, 2018.
29. F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.