Investigating performance of neural networks and gradient boosting models approximating microscopic traffic simulations in traffic optimization tasks

Anonymous Author(s)
Affiliation
Address
email

Abstract

We analyze the accuracy of traffic simulations metamodels based on neural networks and gradient boosting models (LightGBM), applied to traffic optimization as fitness functions of genetic algorithms. Our metamodels approximate outcomes of traffic simulations (the total time of waiting on a red signal) taking as an input different traffic signal settings, in order to efficiently find (sub)optimal settings. Their accuracy was proven to be very good on randomly selected test sets, but it turned out that the accuracy may drop in case of settings expected (according to genetic algorithms) to be close to local optima, which makes the traffic optimization process more difficult. In this work, we investigate 16 different metamodels and 20 settings of genetic algorithms, in order to understand what are the reasons of this phenomenon, what is its scale, how it can be mitigated and what can be potentially done to design better real-time traffic optimization methods.

1 Introduction

Vehicular traffic in cities is a complex process, it involves many components (e.g., vehicles, pedestrians, traffic lights) which are in a constant interaction. Thus, it is not easy to manage it optimally.

In this paper, we analyze an interesting phenomenon which we recently discovered in one of existing traffic optimization approaches, in which neural networks were used as metamodels approximating outcomes of time-consuming traffic simulations in order to speed up the process of finding optimal traffic signal settings [1, 2]. In that approach, neural networks are applied to replace simulations as fitness functions evaluating signal settings in a genetic algorithm (GA), which aims to efficiently find settings minimizing the total time of waiting on a red signal for a given configuration of cars (it is a so called, Traffic Signal Setting problem [4]). It turns out that the accuracy of traffic metamodels based on neural networks, which, in general, is very good in terms of performance on any randomly selected test set, decreases in case of settings found after many iterations of a GA, which makes the optimization process much more difficult. Luckily, genetic algorithms are still able to find relatively good settings (which still make the approach suitable for a real-world traffic management), but their general performance is disrupted because of the aforementioned phenomenon.

In our investigation, we analyzed 8 metamodels based on neural networks and 8 based on LightGBM. First, we trained them on a training set consisting of 85336 evaluations of different traffic signal settings, then, we used them as fitness functions in genetic algorithms aiming to find optimal settings. In total, we investigated 20 configurations of parameters of genetic algorithms to examine how the phenomenon depend on values of different parameters.

The general conclusion is that the phenomenon is universal, although, its scale and properties depend on specific models (e.g., neural networks, LightGBM) and values of hyperparameters (e.g., type of an

activation function). We present the detailed analysis of this phenomenon, its anticipated justification and expected approaches to mitigate its disruptive effect on traffic optimization.

The rest of the paper is organized as follows: Section 2 contains a review of related research works, Section 3 describes our experiments (settings, results and conclusions), Section 4 concludes the paper.

2   Related work

Black-box optimization (in which a black-box, such as simulator, is queried to obtain values for a specific input) is a well-known problem in machine learning [27]. Many approaches have been used, but methods based on Bayesian optimization have become popular in recent years [18], especially when the query is expensive. It is a sequential approach, which tries to find the best point to receive true value in every step. In last years, more advanced methods using Reinforcement Learning [22] and Recurrent Neural Networks [23] appeared - they try to learn how to choose such points. However these methods require incessantly querying, and it is hard to parallelize.

The presented research is inspired by experiments in [1] and [2], in which neural networks were trained to approximate outcomes of traffic simulations and applied to the traffic optimization task (Traffic Signal Setting problem). That approach gave very good results, so we decided to investigate it in details and analyze its performance close to local minima found by genetic algorithms.

3   Experiments

3.1   Setup of experiments

For the purpose of our experiments, we trained 8 metamodels based on neural networks and 8 based on LightGBM. Below, we describe our dataset, metamodels and the genetic algorithm settings.

3.1.1   Dataset

The dataset used for training metamodels was generated using the Traffic Simulation Framework (TSF) [3], similarly as in [1, 2] (TSF is able to run microscopic traffic simulations on realistic maps, consistent with the OpenStreetMap format, using a traffic model based on cellular automata). First, we selected an area in Warsaw consisting of 21 crossroads with traffic signals (Stara Ochota district). Then, we ran 105336 simulations with randomly selected traffic signal settings as an input. Each traffic signal setting can be represented as a vector of 21 elements, each corresponding to a traffic signal offset (being an integer value from the set $\{0,1,\ldots,119\}$) for one of 21 selected crossroads. For each setting, TSF computed the total time of waiting on a red signal in the selected area. For the purpose of experiments (i.e., training metamodels), we divided the set into a training set (consisting of the first 85336 elements) and a test set (the remaining 20000 elements).

We made the dataset publicly available to scientific community to enable a further and more comprehensive research on this topic, the file with the whole dataset is available on the website [28] - each line consists of 21 offsets and the total time of waiting on a red signal computed using TSF for a given setting. It is also possible to get access to a microservice evaluating new settings using TSF, which was used to generate the training set (in that case, we ask for a direct contact with the authors of this paper).

3.1.2   Metamodels based on neural networks

Neural networks are powerful and highly adaptive machine learning algorithms. They are proven to be universal approximators [9] and achieve state-of-the-art results on many tasks despite significant overparametrization [10]. In our experiments, we tested multiple fully-connected network architectures with and without residual connections [11], regularized using $l2$-weight regularization and Batch Normalization layers [12]. Our models were trained using RMSProp optimizer [13], for 200 epochs with the batch size of 128 and a mean squared error as the loss function. In order to normalize the input data and capture its periodic properties we transformed each coordinate x of the input data to a pair $\left(\cos\frac{2\pi x}{120}, \sin\frac{2\pi x}{120}\right)$, what changed the input shape from $(21,)$ to $(42,)$. We trained 216 models with different values of regularization rates, number of units and layers, activations (ReLU [14] and tanh [15]) and chose a set of 8 models (from the set of the best 30 models with the lowest relative

error rates) which had the greatest variability in terms of model metaparameters. All models were trained using Keras [16] package with Tensorflow [17] backend. The parameters of choosen models are presented in Table 1 in the Supplementary material 1 [24].

3.1.3 Metamodels based on LightGBM

Gradient boosting models belong to a family of machine learning algorithms, where the main idea assumes that weak predictive models become better with subsequent iterations, throughout learning from mistakes done by previous models. In our case, weak learners are represented by decision tree regressors. The most widely used implementations of this algorithm are XGBoost [5] and LightGBM [6]. The LightGBM algorithm is a gradient boosting framework, which uses tree based learning algorithms. In contrast to other tree based algorithms, which use level-wise tree growth, LightGBM grows trees vertically. It has been proven that LightGBM remarkably outperforms XGBoost in terms of memory consumption and computational speed [6], which agrees with the outcomes of our experiments. These factors are vitally important in our research, as the execution time of a traffic metamodel should be as short as possible. On top of that, we managed to achieve slightly better accuracy with LightGBM, hence our choice is to base the further experiments upon this gradient boosting algorithm.

In order to find good values of hyperparameters, the Tree-structured Parzen Estimator was used, which is an efficient and robust hyperparameter optimization technique [7]. LightGBM has many parameters[1], we initially selected a core set of hyperparameters and found their optimal values through training and validating the model on our training set, with 20% split between train/test sets (see Table 2 in the Supplementary material 1 [24]).

Our idea was to create 8 different models and analyze how they perform comparing to real TSF's outcomes, trying also to find some patterns in their behaviour. We decided to keep optimized values of aforementioned hyperparameters and experiment with some other important parameters. Since LightGBM uses a leaf-wise growth algorithm, num_leaves (number of leaves) controls its complexity. Regression_l1 evaluates a model based on the mean absolute error metric, regression_l2 and Poisson use root mean squared error and negative log-likelihood, respectively, as an evaluation metric. Apart from traditional Gradient Boosting Decision Tree (gbdt) method, we also included Dropouts meet Multiple Additive Regression Trees (dart) method in our experiments. Different values of num_leaves, regressions and boosting techniques influence results strongly, so we decided to play with these parameters, train multiple models and select best 8 models. Table 3 in the Supplementary material 1 ([24]) presents the selection of chosen models along with relative errors made on our test set.

3.1.4 Settings of genetic algorithms

The investigated Traffic Signal Setting problem belongs to the family of nonconvex combinatorial optimization problems and for such tasks it is often difficult to find global optimum and even finding suboptimal solutions may be challenging, especially when obtaining value of the utility function evaluating traffic signal settings is also computationally expensive and that's our case. Usually, the best approach is to use metaheuristics [8] - algorithms searching space of possible solutions, which make decisions which points to evaluate next based on past experience, e.g., genetic algorithms, simulated annealing, Bayesian optimization, tabu search, particle swarm optimization. We conducted experiments with these algorithms and realized that genetic algorithms outperforms other methods in our task. Therefore, we decided to focus in next experiments only on genetic algorithms.

In our use case, we represented traffic settings as 21-element vectors of offsets of traffic signals, being integer numbers from the set {0,1,2,...,119} (e.g. [4,4,48,82,41,57,35,54,54,14,30,31,31,31,92,111,13,13,77,110,4]) and we evaluated them using fitness functions representing computed (by TSF) or estimated (by neural networks or LightGBM models) total time of waiting on a red signal.

For conducting experiments using genetic algorithms, we developed a dedicated Python library. Thanks to that, we were able to test 504 configurations of hyperparameters of our algorithms (all configurations are described in the Supplementary Material 2 [25]). For each configuration, we ran

---

[1]LightGBM's parameters are explained on the website: `https://lightgbm.readthedocs.io/en/latest/Parameters.html`

experiments for each of our 16 models, 5 times per each model (each run starts from a random initial population, so we wanted to test different initial settings), giving 2520 runs for each model.

We had observed earlier that evaluating genotypes using TSF takes about 30 seconds, which is too long, but in case of evaluations using metamodels (neural networks and LightGBM) the time of inference was much lower (less than 0.1 second), so we were able to conduct such compute-intensive experiments.

3.2 Results of experiments

3.2.1 Initial experiments

In the past experiments, genetic algorithms prove to be very efficient in finding suboptimal solutions for the Traffic Signal Setting problem ([1], [2]), so it was no surprise that they were able to find good solutions in our case (at least, according to evaluation using metamodels). Usually, genetic algorithms stopped converging after about $50 - 100$ iterations (we had realized that earlier, so we decided to set 100 as the number of iterations). The average value of the fitness function (computed using TSF) in the training set was about 48923, but genetic algorithms were able to find points with a value at the level $32000 - 33000$ according to metamodels - as was later realized, the optimal values returned by genetic algorithms usually differed from TSF outcomes more than the average error of metamodels (about $1.7\% - 2\%$), but settings found by genetic algorithms were still relatively good and much better than randomly selected settings.

Since we realized that the accuracy of approximations using metamodels is worse near local optima, we decided to examine this phenomenon. For that purpose, we had to compute true values of simulations for a large number of traffic signal settings. Each run of a genetic algorithm produced about 120 (number of setting in a population) x 100 (number of iterations) = 12000 settings (in some cases the number was lower, because some settings were shared among different populations) and since we wanted to investigate universality of the phenomenon and test different values of hyperparameters of metamodels and genetic algorithms, we had to carefully select which settings should be evaluated using TSF (a single evaluation using TSF took about 30 seconds on standard machines, which may give large time of computations in case of large number of evaluated settings).

For each of 16 models, from 2520 genetic algorithm runs we selected best 10 (according to the final best setting) and additional 10 randomly from 90 runs on positions $11 - 100$ (according to best found setting using a given model), so later we were investigating only 320 GA runs and for each run we decided to evaluate using TSF 100 trajectory settings (best settings from each one of 100 iterations) and 100 best settings found by a genetic algorithm run across all iterations.

3.2.2 Distribution of errors

For each model, we analyzed distributions of errors of approximations for best 100 settings found in each of selected 20 genetic algorithm runs (which gives 2000 points, in total) - the summary is on the Figure 1 presenting relations between approximations and true values returned by TSF.

One of interesting conclusions is that in most cases metamodels underestimate results of simulations and usually the error of approximation is larger than the average error of a given model on a random test set ($1.7 - 2\%$) and it can reach 12.9% (in 14 out of 16 models it is larger than 2.16%), the same happens to the maximal error which can reach 21.5%.

Another interesting observation is that in case of neural networks activation function may significantly change the distribution of an error. For ReLU, neural network almost always underestimate results of simulations, for tanh the error is more symmetric.

We also analyzed errors for settings from GA trajectories and found that in case of last settings (which are also expected to be best) errors of approximations increase (Supplementary material 3 [26]).

3.2.3 Trajectories

We also analyzed errors for traffic signal settings from trajectories (Figure 2). It confirms that, in general, error of approximations increases toward underestimating outcomes of simulations. Also, in case of LightGBM errors are becoming greater. A very important fact is that despite of increasing errors, genetic algorithms are still able to find much better settings than in the initial populations.
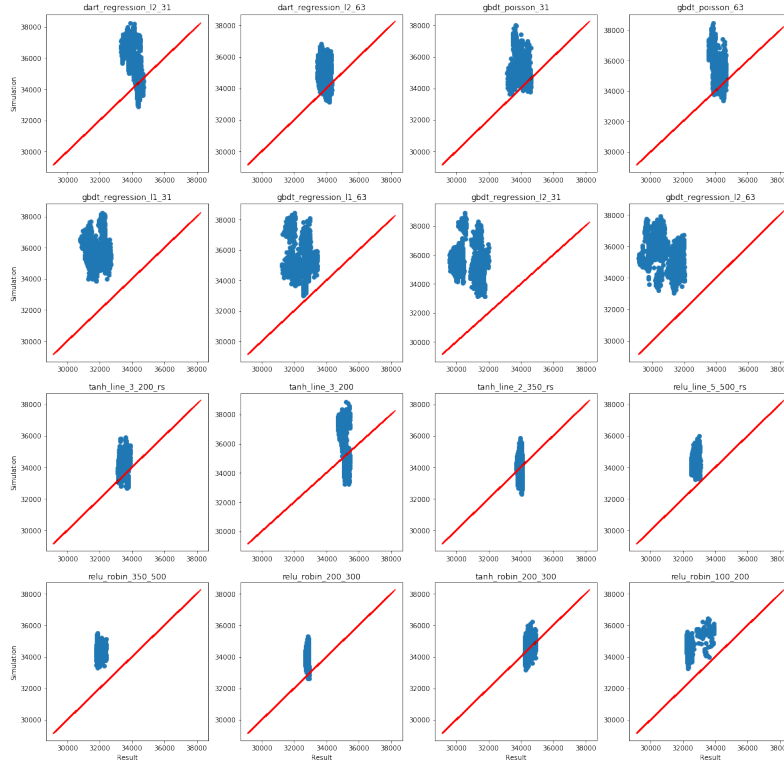
4

Figure 1: Comparing model prediction and simulation for settings with the lowest predictions (100 best settings for 20 genetic algorithm runs) for all 16 investigated models.
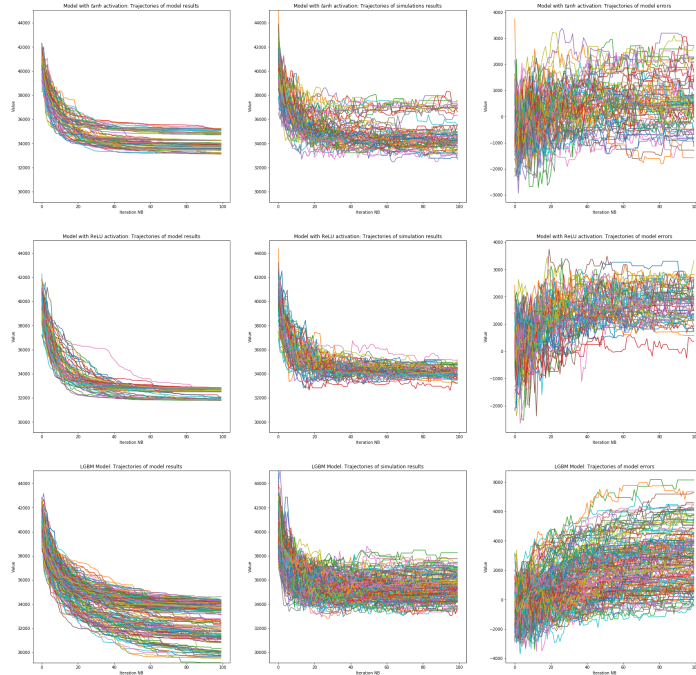

Figure 2: Trajectories of GA runs, values of simulations and errors for neural networks with tanh (first row) and ReLU (second row) activation functions and for LightGBM (third row).

Figure 3: Principal component analysis on best settings from different populations in GA
.

3.2.4 Where do genetic algorithms converge to?

We also ran a principal component analysis [21] on settings from trajectories and found out that, depending on the metamodel, genetic algorithms may converge to totally different points in the space of possible solutions (Figure 3).

3.3 Discussion of results

Based on our analysis, we conclude that for traffic signal settings found by genetic algorithms, errors of approximations increase and the phenomenon is universal - it can be observed in case of many different metamodels (neural networks, LightGBM), but its scale and properties may differ depending on some parameters, e.g., activation function in case of neural networks. We see 2 potential reasons of this phenomenon:

- underrepresentation of local minima found by genetic algotihm in the training set

- the fact that genetic algorithms seek local minima and, therefore, they may also find points for which values of approximation are minimal and far from real simulation results

4 Conclusions and future work

In the paper, we investigated performance of metamodels (based on neural networks and LightGBM) approximating outcomes of traffic simulations. Metamodels were used as fitness functions in the traffic optimization task and we found out that genetic algorithms searching for optimal settings of traffic signals, end up in areas of subspaces of possible settings, in which distribution of the error of approximation is much different than the standard distribution on a test set (errors are larger), which makes the traffic optimization task more difficult. We carried out detailed analysis of this phenomenon and it turns out that some hyperparameters of metamodels (e.g., activation function) have different impact on results. Moreover, the principal component analysis showed that genetic algorithms may converge to totally different areas of the searched space, depending on metamodels.

We also started analyzing how this phenomenon, which disrupts the traffic optimization task, can be mitigated. We see at least 2 interesting research directions:

- active learning / transfer learning approach - adding to the training set new settings found by a genetic algorithm (we investigate methods which may help make decisions when to run a time-consuming simulation in order to evaluate new points) and retraining metamodels

- ensemble learning approach - training several different models and taking as an output the average outcome of some of these models

We've already started working on these topics, but we also invite the scientific community to join this research (which is especially important from the traffic optimization perspective) and that's why we share a dataset which was used in our research [28].

References

[1] Gora, P. & Kurach, K. (2016) Approximating Traffic Simulation using Neural Networks and its Application in Traffic Optimization, in *NIPS 2016 Workshop on Non-convex Optimization for Machine Learning: Theory and Practice*.

[2] Gora, P. & Bardoński, M (2017) Training neural networks to approximate traffic simulation outcomes, in *5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), IEEE*, pp. 889—894.

[3] Gora, P. (2009) Traffic Simulation Framework - a Cellular Automaton based tool for simulating and investigating real city traffic, in *"Recent Advances in Intelligent Information Systems*, pp. 641-653.

[4] Gora, P. & Pardel, P. (2015) Application of genetic algorithms and high-performance computing to the Traffic Signal Setting problem, in *24th International Workshop, CS&P 2015, Vol. 1, Rzeszów*, pp. 146-157.

[5] Chen, T. & Guestrin, C. (2016) XGBoost: A Scalable Tree Boosting System, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794.

[6] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. & Liu, T. (2017) LightGBM: A Highly Efficient Gradient Boosting Decision Tree, in *Advances in Neural Information Processing Systems 30*.

[7] Bergstra, J., Bardenet, R., Bengio, Y. & Kegl, B. (2011) Algorithms for Hyper-Parameter Optimization, in *NIPS'11 Proceedings of the 24th International Conference on Neural Information Processing Systems*, pp. 2546–2554.

[8] Luke, S. (2015) Essentials of Metaheuristics, in $https: // cs. gmu. edu/ ~sean/ book/ metaheuristics$.

[9] Cybenko, G. (1989) Approximations by superpositions of sigmoidal functions, in *Mathematics of Control, Signals, and Systems, 2(4)*, pp. 303–314.

[10] Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y. & Lacoste-Julien, S. (2017) A closer look at memorization in deep networks, in *Technical Report Preprint: arXiv:1706.0539*.

[11] He, K., Zhang, X., Ren, S. & Sun, J. (2015) Deep residual learning for image recognition, in *Technical Report, arXiv:1512.03385*.

[12] Ioffe, S. & Szegedy, C. (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift, in *arXiv:1502.03167*.

[13] Ruder, S. (2016) An overview of gradient descent optimization algorithms, *arXiv:1609.04747*.

[14] Hahnloser, R., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J. & Seung, H.S. (2016) Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit, in *Nature. 405*, pp. 947–951.

[15] Glorot, X. & Bengio, Y. (2010) Understanding the difficulty of training deep feedforward neural networks, in *International Conference on Artificial Intelligence and Statistics (AISTATS'10), Society for Artificial Intelligence and Statistics*.

[16] Chollet, F. et al., Keras, $https: // github. com/ fchollet$.

[17] Abadi, M., et al. (2016) TensorFlow: Large-scale machine learning on heterogeneous systems *arXiv* $https: // arxiv. org/ abs/ 1603. 04467$. Software available from http://tensorflow.org.

[18] Brochu, E., Cora, V.M. & de Freitas, N. (2010) A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning, in *arXiv* $https: // arxiv. org/ abs/ 1012. 2599$.

[19] Driessen, L., Brekelmans, R., Hamers, H. & den Hertog, D. (2006) On D-optimality based trust regions for black-box optimization problems", in *Struct Multidisc Optim*, pp. 31-40.

[20] Rashmi, K.V. &, Gilad-Bachrach, R. DART: Dropouts meet Multiple Additive Regression Trees, in *arXiv* $https: // arxiv. org/ abs/ 1505. 01866$.

[21] Pearson, K. (1901) On Lines and Planes of Closest Fit to Systems of Points in Space, in *Philosophical Magazine, 2 (11)*, pp. 559-–572.

[22] Li, K. & Malik, J. (2017) A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning, in *arXiv* $https: // arxiv. org/ abs/ 1703. 00441$.

[23] Chen, Y., Hoffman, M.W., Colmenarejo, S. G., Denil, M., Lillicrap, T. P. & de Freitas, N. (2017) Learning to Learn for Global Optimization of Black Box Functions, in *arXiv* $https: // arxiv. org/ abs/ 1611. 03824v1$.

[24] Supplementary material 1 - Neural networks and LightGBM models https://1drv.ms/b/s!Amty7h2ZkQOLksQcLId46YXoB3fZIw.

[25] Supplementary material 2 - Genetic algorithm settings `https://1drv.ms/f/s!Amty7h2ZkQOLksQf0NpVKApQVak37g`.

[26] Supplementary material 3 - distribution of errors on genetic algorithm trajectories `https://1drv.ms/b/s!Amty7h2ZkQOLksQkf2ybcgaqxOlzvw`.

[27] Amaran, S., Sahinidis, N.V., Sharda, B., & Bury S.J. (2014) Simulation optimization: a review of algorithms and applications, in *4OR-Q J Oper Res, vol. 12, iss. 4*, pp. 301-333.

[28] Dataset used for experiments: `https://1drv.ms/u/s!Amty7h2ZkQOLksQg_EaezOC6DkdgEQ`.