

Estimation of hidden driver properties based on the driving behavior

Pascal Held **Rudolf Kruse**

Otto-von-Guericke University of Magdeburg
Faculty of Computer Science
Department of Knowledge Processing and Language Engineering
Universitätsplatz 2, D-39106 Magdeburg
Tel.: +49 391 67 18718
Fax: +49 391 67 12018
E-Mail: {pheld,kruse}@iws.cs.uni-magdeburg.de

Abstract

Given a better knowledge about the driver the car's driver assistance systems could be specifically adapted, potentially increasing the systems' acceptance. Our purpose is to estimate hidden driver properties underlying the driving behavior. We used a linear combination of different quantities as basis of the similarity calculation. With the aid of an evolutionary algorithm we determined these quantities as well as their weights. In a study with 26 subjects three hidden target properties were examined and estimated. We reached a correlation of approximately 0.7 between the predicted values and given measurements. In order to exclude overfitting of the model, an additional optimization was carried out using random values. In this case only a considerably smaller correlation of 0.36 could be achieved. As a consequence we observed that the basic assumption applies at least to the three examined target properties. We conclude that an estimation based on driving behavior is possible.

1 Introduction

In modern vehicles the incorporation of assistance systems is constantly increasing. Their goals are, among other things, to reduce accident consequences or completely avoid them altogether. To do so, the system activates the brake for example. The greatest challenge these systems face is the decision when to intervene. An early intervention can increase the efficiency of the system. This results, however, in drivers more frequently perceiving the activation as necessary. Given a better knowledge about the driver, the systems could be specifically adapted, potentially increasing the systems' acceptance. Such an estimation of the driver is the goal our work. The main idea how to estimate from given data is explained in Section 2. Our purpose is to estimate hidden driver properties underlying the driving behavior. This estimate is based on the assumption that drivers with similar properties behave similarly concerning certain quantities. To select which properties to use, we developed an evolutionary algorithm, which is described in Section 3. We tested our algorithm with different data sets in Section 4. The results obtained could be used to for example parameterize assistance systems. A similar approach was presented from Bauer et al. in [1].

2 Methods

2.1 Used Data

We used data from two different sources for our analysis. The first dataset we examined was recorded from a driving simulator, where 21 test persons passed both a city and a country road scenario. Additionally one target property was determined for every person. As a second source we used a field study with 26 test persons. Every journey was split into three parts, i.e. freeway, country road, city scenario. For every test person three target properties were determined [2].

While driving we collected different driving parameters every 100 ms. These parameters were distance to the vehicle driving ahead, acceleration, gas pedal value, velocity, v and cross acceleration.

The distance was detected with the help of a radar system. The other parameters were directly derived from the car's CAN bus. We transformed this distance into three measures, i.e. the absolute distance d_{abs} in meter, the time gap Δt in seconds, and the time to collision TTC in seconds. [3]

$$\Delta t = \frac{d_{abs}}{v_{own}} \quad (1)$$

$$TTC = \frac{d_{abs}}{v_{own} - v_{ahead}} \quad (2)$$

During the whole driving session we can distinguish the following situations: free driving (no car ahead), driving behind a car, free stopping (no car ahead), stopping behind a car, free start (no car ahead), starting behind a car and global behavior.

Stopping processes were detected by velocity analysis.

2.2 Generating Parameters

The question is how to generate features from collected data. One requirement is that there should be a cumulative evaluation of the previous driving session. We used statistical measures such as mean value, variance, and percentage above a given threshold. These are measures which are easy to update online. In our work we used about 40 different characteristic values from all possible combinations.

We normalized all values to the unit interval using the percentage rank method, where $f_{cum}(x_v)$ is the cumulative frequency of the value x_v or lower and N is the size of the sample size. [4]

$$PR_v = 100 \cdot \frac{f_{cum}(x_v)}{N} \quad (3)$$

This was necessary because we compared different parameters like velocity and acceleration. So for every feature we obtained a uniform distribution in the unit interval.

2.3 Estimating Target Values

To estimate the target values we used a case-based reasoning approach [5]. Comparing the features of the current driving session to the values in the reference data base allows us to compute the target value with a k -NN-algorithm with $k = 3$ as a weighted average, whereas the weights represent the similarity of the current driving sessions to the three most similar reference tracks.

The task is how to find similar reference tracks. We used a selection of some of the features by using a similarity measure based on the euclidean distance. But not every feature is equally significant. So, we weighted the different dimensions as well. To determine the selected features and their weights we used an evolutionary algorithm [6] which is presented in the next section.

3 Evolutionary Algorithm

We started with an initial population of 50 random candidate solutions. An exact description of these individuals follows in section 3.2. The following generations contain 10, including the three best, unchanged individuals from the previous generation, 20 individuals generated by mutation and 20 individuals generated by crossover. The algorithm terminates after 100 generations.

3.1 Representation of the Individuals

We represent the individuals as a vector of double values. Every field is related to the weight of one feature. We want a bijective mapping between individual and weights configuration. Therefore, we normalize all vectors so that the sum of all weights is 1. This was necessary to avoid unexpected results on crossover with equal individuals with a different scaling.

The value 0 of a field therefor indicates that this feature is not used.

3.2 Initial Population

For the initial population we generated 50 random individuals. The generation is based on a standard normal distributed random variable Z . Every field in the individual's vector is calculated in the following way.

$$\forall i \leq N, i \in \mathbb{N} : g_i = \frac{1 + Z}{2} \quad (4)$$

Normally, the sum of all fields is not 1, so the generated vector must be normalized. Values lower than 0 will be set to 0.

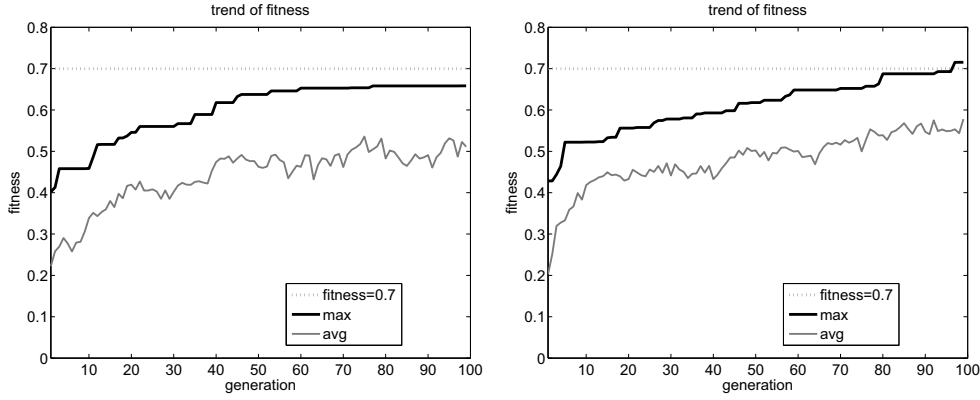


Figure 1: left: evolution of the fitness with standard roulette wheel method, right: using power law scaling with $k = 1.005$

3.3 Fitness Function

To assign a fitness value to the individuals we made an experiment with every single individual. We used the leave-one-out method [7] to estimate the target value based on the other reference tracks with the weights given by the individual. After this we calculated the Pearson correlation coefficient r between the estimated target value and the reference value. The goal is to maximize the correlation between the estimated and the given values.

To avoid overfitting we introduced a penalty term which punishes individuals that use many features. Altogether we derived the following fitness function

$$f = r \cdot \left(1 - \frac{1}{100}n\right), \quad (5)$$

where n is the number of features with a weight greater than 0.

3.4 Selection Method

To select random individuals we used the roulette wheel method [6]. Additionally the three best individuals were part of the next generation without any change, the so-called elitist strategy. This guarantees that the best individuals will not get lost during the optimization [8]. During the optimization there can be individuals with a fitness values less than 0. To prevent an inclusion of such individuals in the roulette wheel method, a minimal fitness of 0.01 was assigned to these individuals.

The left part of Figure 1 shows the evolution of the fitness with the standard roulette method. You can see an increasing fitness value during the first 50 steps of the optimization. Afterwards, only small improvements can be observed. This may be caused by the selective pressure becoming too low. To avoid this we tried different scaling methods for the fitness values, especially the power law scaling [9]

$$f^* = f^k. \quad (6)$$

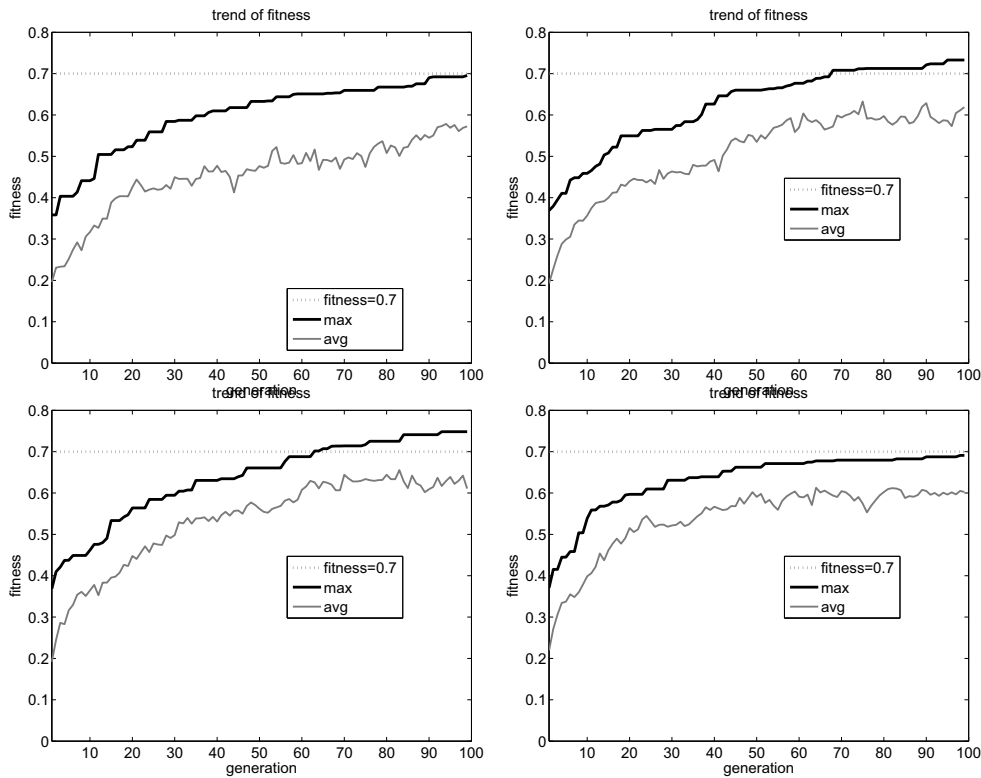


Figure 2: evolution of the fitness with different values of c (top left: $c = 2$, top right: $c = 3$, bottom left: $c = 4$, bottom right: $c = 5$)

A commonly used parameter is $k = 1.005$. The right part of Figure 1 shows the evolution of the fitness with the applied power law scaling with $k = 1.005$. It is easy to see that this method outperforms the standard roulette method in our case. In addition to a constant value for k it is possible to change this value over time. In our work we tried an approach based on the fitness of the best individual of the population

$$k = c * \max(f). \quad (7)$$

We tested different values of c . The results are presented in Figure 2. The best results were obtained when using $c = 3$ or $c = 4$. For higher values, e.g. $c \geq 5$, the influence of the fitness is too strong in the starting phase of the optimization. The optimization gets stuck in a local optimum and no further evaluation of the search space is performed. On the other hand, small values slow down the convergence of the population and do not lead to a significant improvement compared to standard power law method. In the following, we set $c = 3$ because this value allows a good evaluation of the search space and a good evolution of the population.

3.5 Genetic Operations

There are two types of genetic operations we used in our algorithm, i.e. mutation and crossover. As crossover operation we used the standard uniform crossover [10].

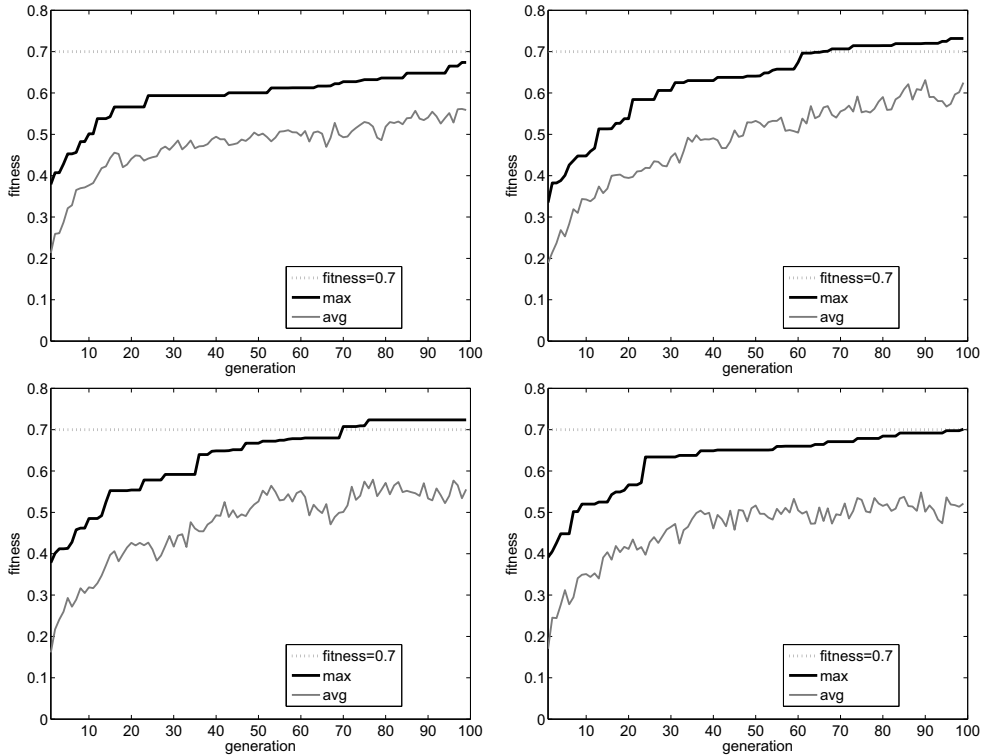


Figure 3: Evolution of the fitness with different k (top left $k = 0.1$, top right $k = 0.2$, bottom left $k = 0.3$, bottom right $k = 0.4$)

The other genetic operation we used is mutation. The aim of the mutation is to change some values of the individuals randomly to explore the search space in the neighborhood of existing individuals. To achieve this we add a standard normal distributed value X to the previous values g_i

$$g_i^* = g_i + k \cdot X. \quad (8)$$

In Figure 3, we show the evolution of the fitness for different values of k . For $k = 0.1$, there is a slow increasing of the fitness. This could be due to attempts to reach local optima. For higher values, e.g. $k = 0.4$, there is a fast increasing of the value. This means good areas are found very quickly. But high mutation values prevent reaching the optima in these areas. It is therefore good to take values around $k = 0.2$ and $k = 0.3$. These values have similar evolutions of the fitness. We used $k = 0.2$ because local optima were easier reachable.

In Figure 4 we show the evolution of the fitness for different mutation probabilities p . The closer the average of the fitness is at the maximum, the less strong the search space is investigated. This happens because many individuals are near local optima. Premature convergence is an indication that a lot of individuals are stuck in local maxima.

For small values of $p \leq 2/n$, the search space is examined poorly. High values, e.g. $p = 6/n$, cause divergences of the individuals. Good values are reached in a random way, but they do not tend to really optimize the target function. Rather, individuals near new local optima are found. A higher mutation probability increases this effect.

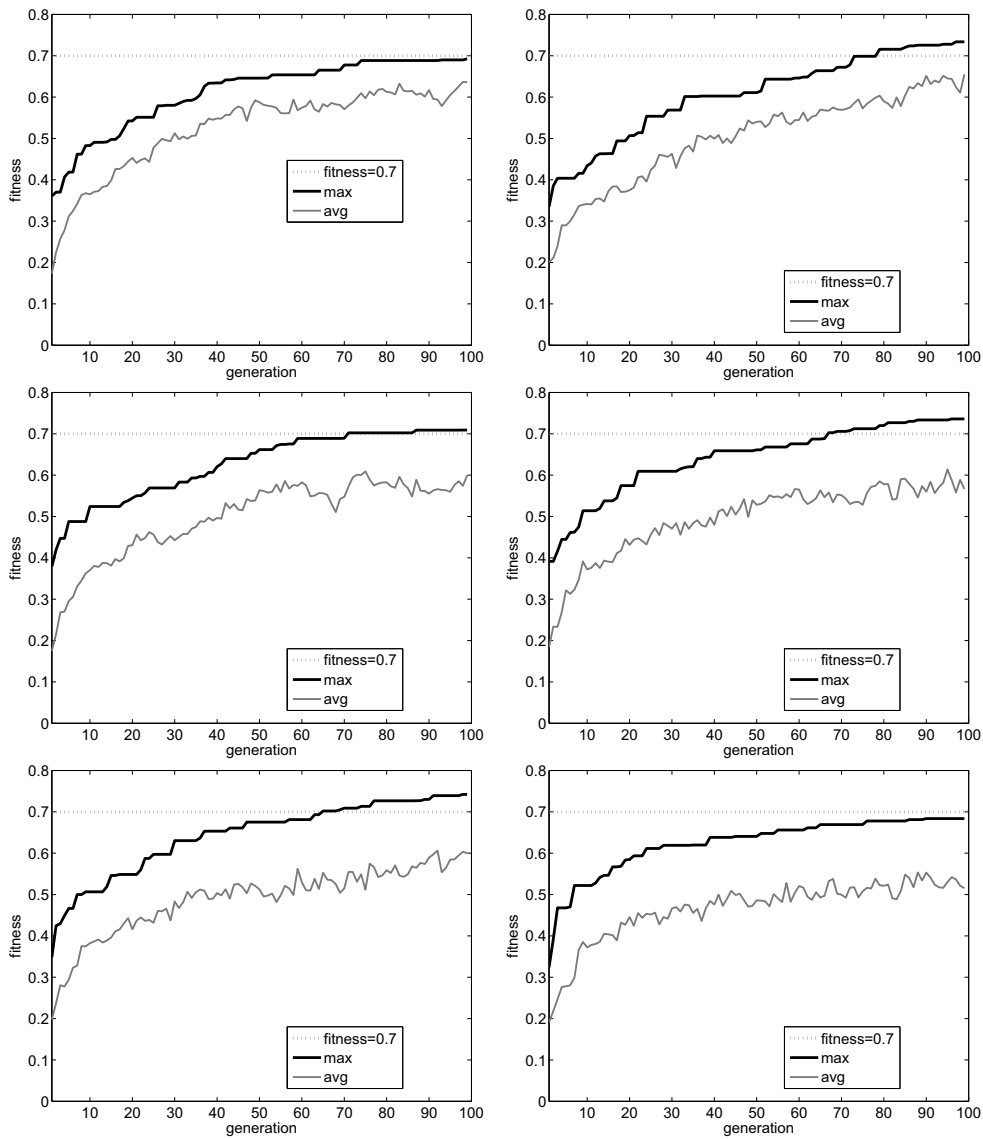


Figure 4: Evolution of the fitness with different mutation probability p (top left $p = 1/n$, top right $p = 2/n$, center left $p = 3/n$, center right $p = 4/n$, bottom left $p = 5/n$, bottom right $p = 6/n$)

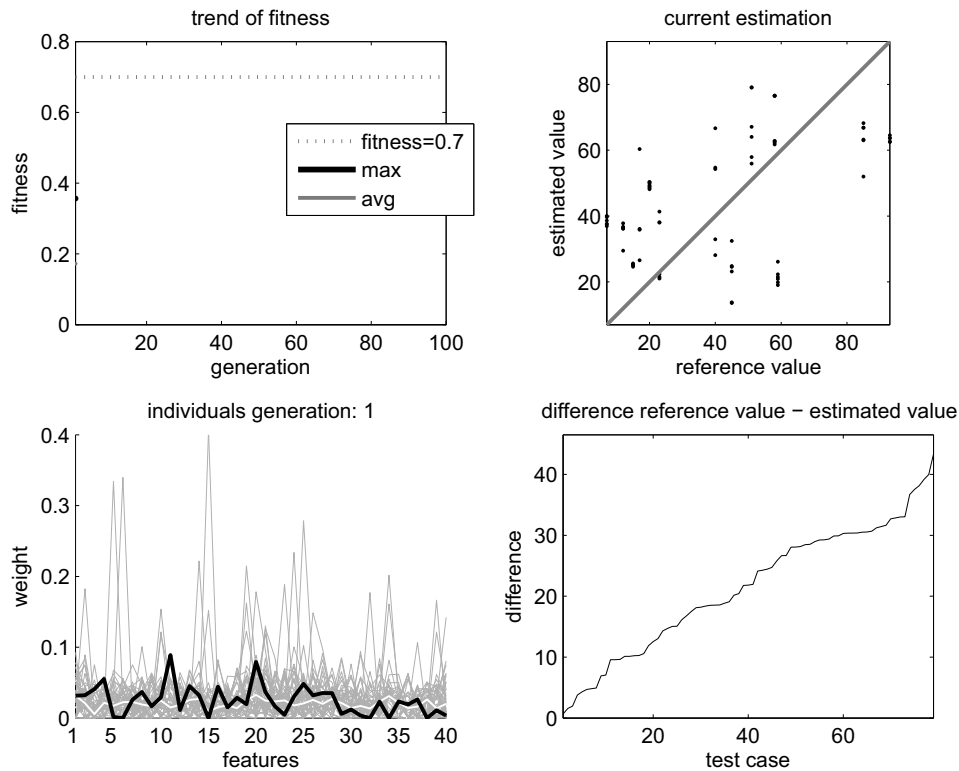


Figure 5: Generation 1 - simulation study

We used $p = 4/n$ as mutation probability as it performs significantly better in our case than $p = 1/n$, which is often used.

4 Results

We evaluated the algorithm with both the data of the simulation and the field study. We took snapshots of different timestamps during the driving sessions. In Figure 5 we show the start situation of the population with the simulation data. In the top left part of Figure 5 we can see the evolution of the fitness over time. In this figure the graph is nearly empty because we are in iteration 0. On the top right, we show the reference value and the estimated value for every single reference measure. This estimated value is based on the reference tracks of the other drivers. The gray line is the optimal fit line. In this iteration the estimation fits the reference tracks poorly. In later iterations there are more data points on the same place. To get a better feeling of the estimated points we calculated the difference of the reference and the estimated value. We ordered these values and plotted them in the bottom right subfigure. In the bottom left, we visualize the population itself. Every single gray line represents one individual. It is easy to see that there are many different configurations. The black line represents the best individual of the current iteration. In table 1 we give some links to online videos of the simulation. In these videos we present the whole evolution of all the experiments.

After 20 generations the estimation results are better. You can see this in Figure 6. This also results in increasing fitness values. The individuals are less scattered.






study	link	QR-code
simulation study	http://youtu.be/ Zm1Y_2wYM9M	
field study - value A	http://youtu.be/ fbLchwuCS_g	
field study - value B	http://youtu.be/ ZvPaEwfpnRU	
field study - value C	http://youtu.be/ uL-N5uszclM	
random data	http://youtu.be/ cmernihK1vHV4	

Table 1: links to online videos of the simulation

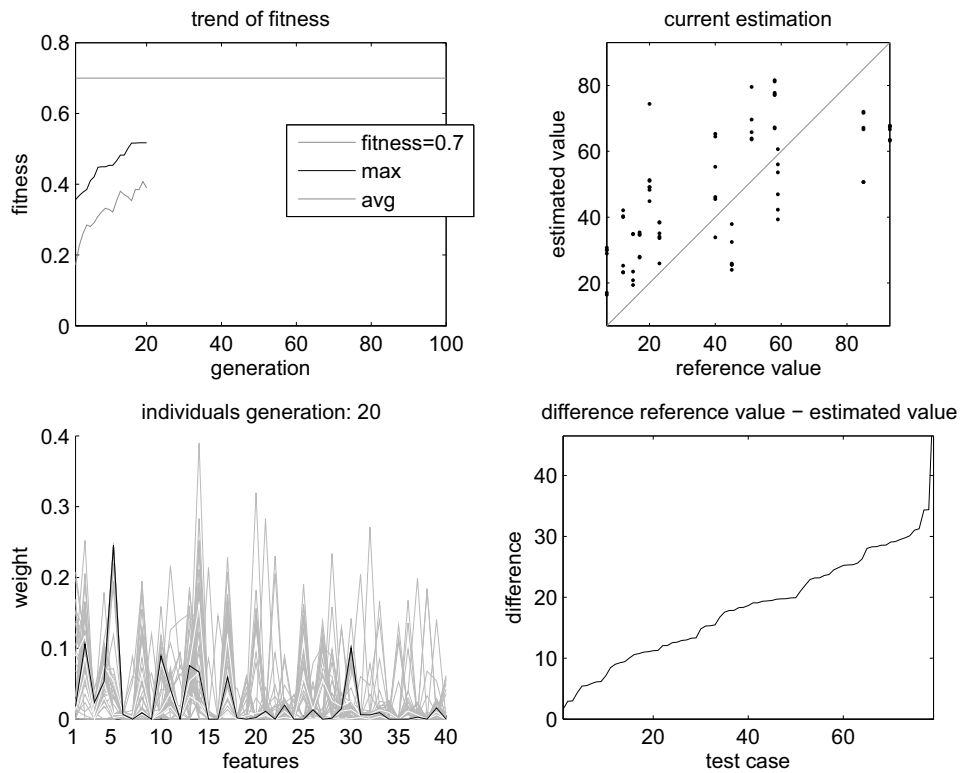


Figure 6: Generation 20 - simulation study

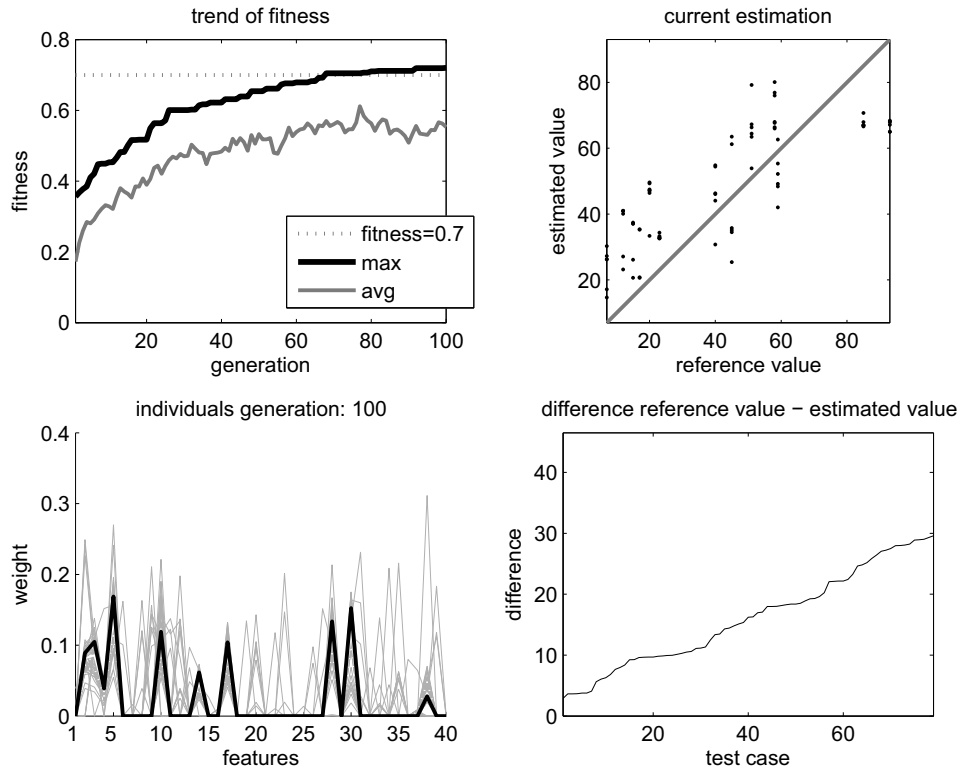


Figure 7: Generation 100 - simulation study

After 100 generations the algorithm terminates. The results are shown in Figure 7 and one can see that the estimation is much better. The individuals of the population are similar to each other, but not all individuals are near the best individual. So, the search space was still explored properly. We finished with a fitness of 0.72 and 10 selected features. If we leave out the penalty term for using too many features, the correlation will even rise to a value of 0.80:

$$correlation = \frac{fitness}{1 - \frac{\#features}{100}} = \frac{0.72}{1 - \frac{10}{100}} = 0.80. \quad (9)$$

In the field study we get similar results. In Figures 8 and 9 one will see the evolution of the fitness for the first target value. We finished with a fitness value of 0.62 and a feature set size of 13. This results in a correlation of 0.71. In Figures 10 and 11 we present the final states for both other target values. For the second target value we still achieved a fitness value of 0.49 with 12 selected features and for the third target value we computed a fitness value of 0.62 with 9 selected features. These results yield correlations of 0.56 and 0.67 respectively.

To verify that these results are not based on overfitting, we tested a target value based on random data. Every reference track was assigned to a uniformly distributed value between 0 and 100. In Figure 12 we show the results for this test. We got a fitness value of 0.3 with 16 selected features. This result yield a correlation of 0.36.

We tested the developed algorithm on different data sets. Good results were obtained on real target values. In contrast random target values resulted in poor results, which thus

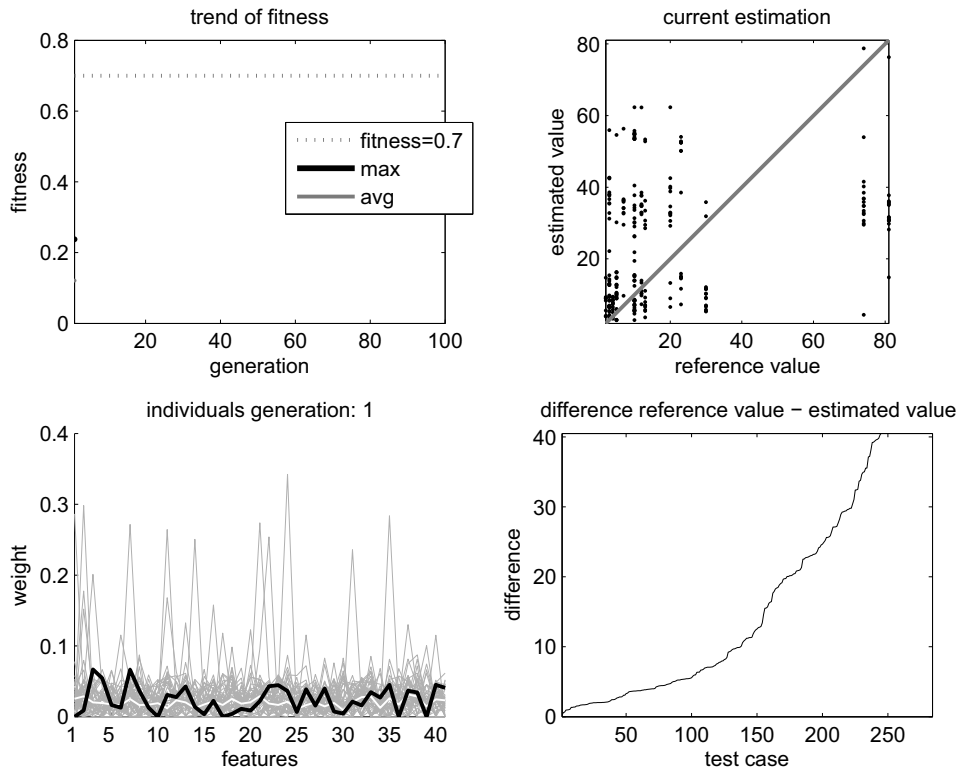


Figure 8: Generation 1 - field study target value A

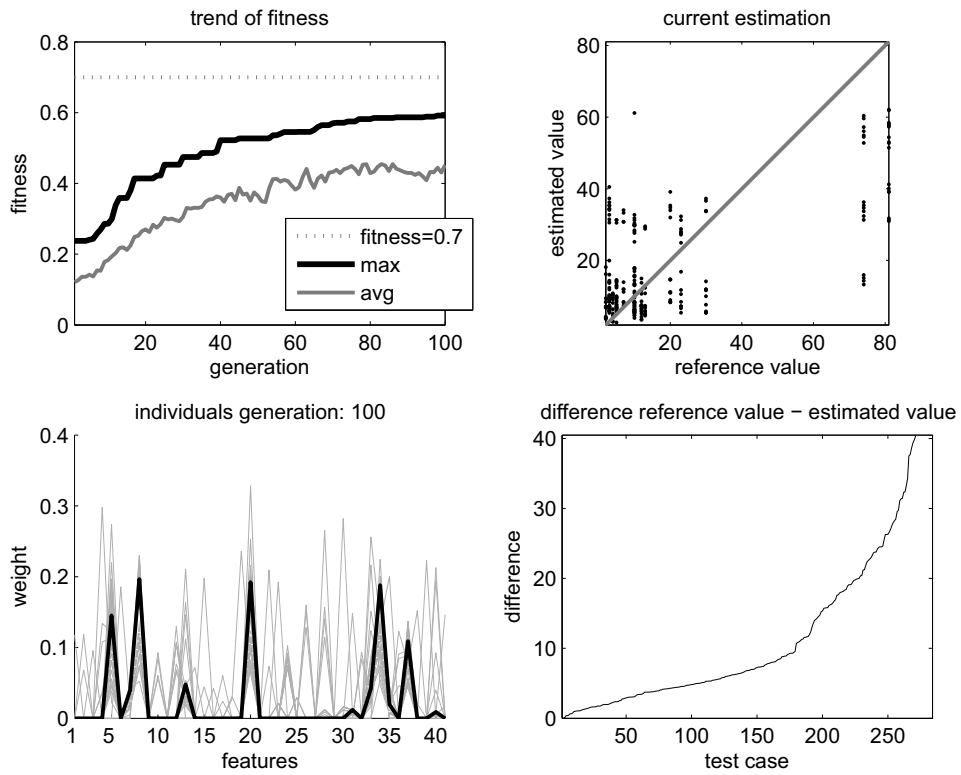


Figure 9: Generation 100 - field study target value A

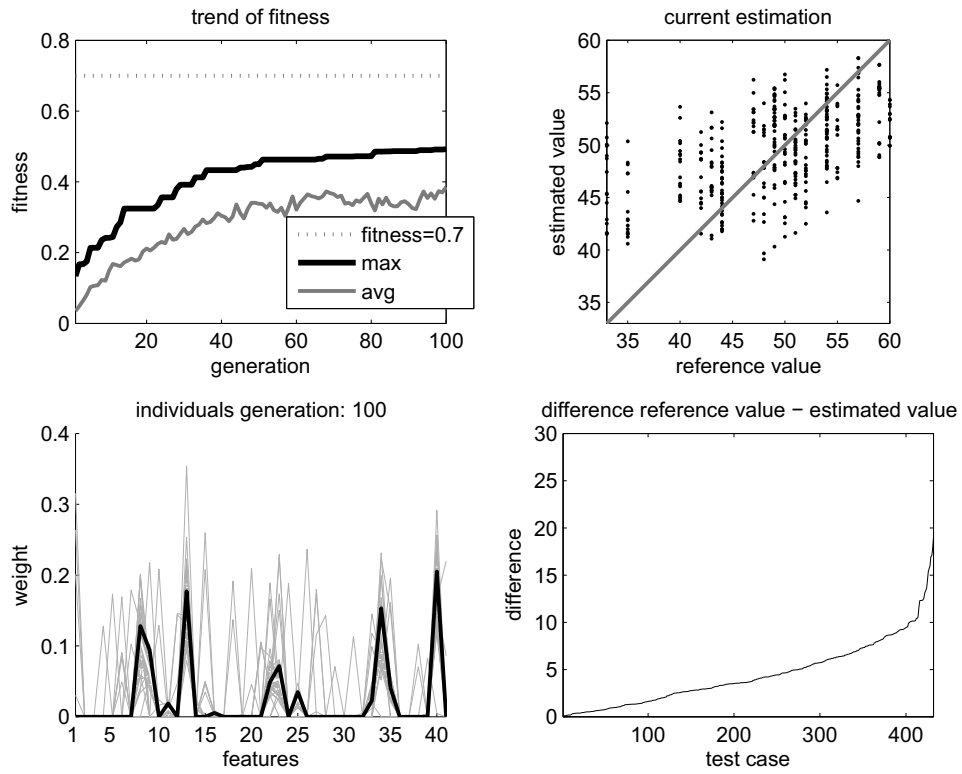


Figure 10: Generation 100 - field study target value B

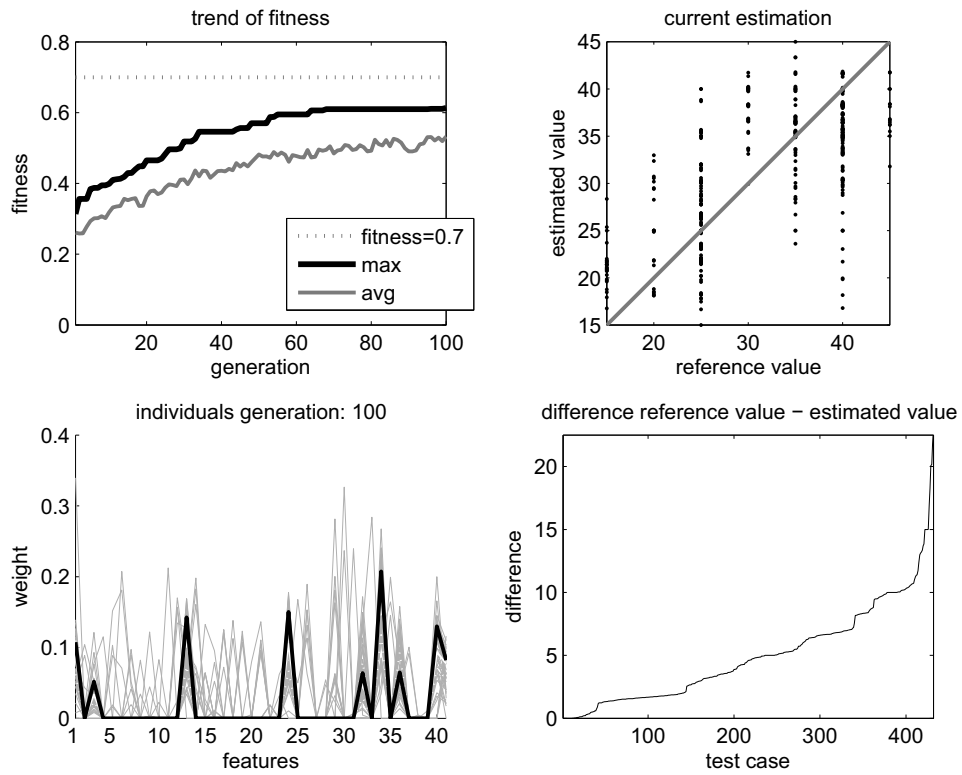


Figure 11: Generation 100 - field study target value C

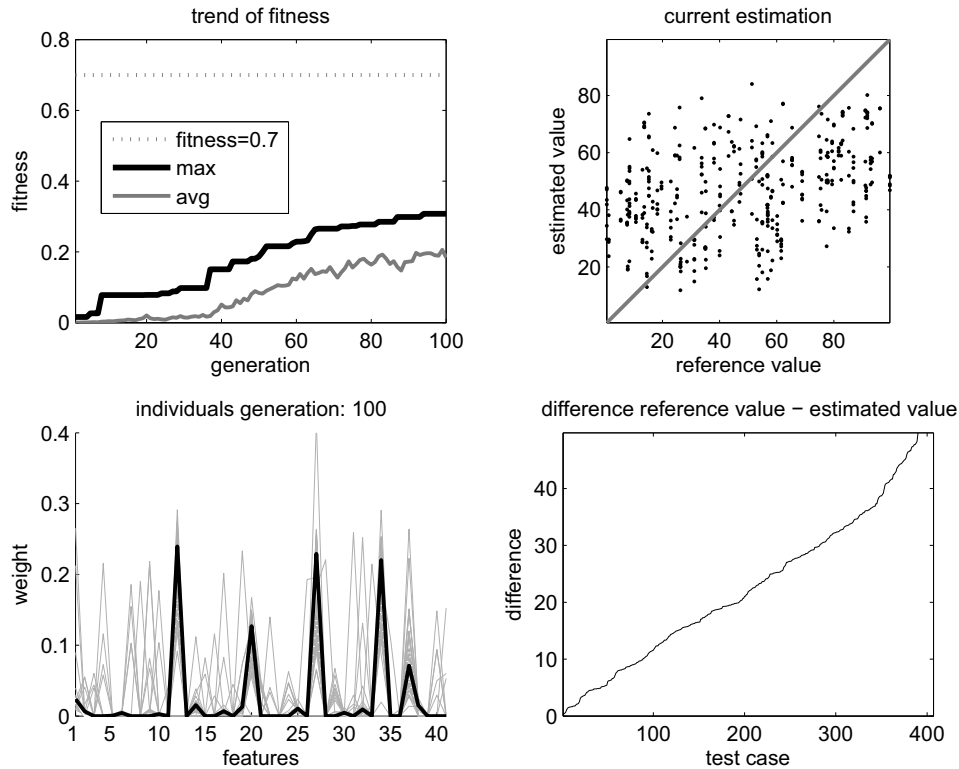


Figure 12: Generation 100 - random data set

study	max. fitness	#parameter	correlation
simulation study	0.72	10	0.8
field study - value A	0.62	13	0.71
field study - value B	0.49	12	0.56
field study - value C	0.62	9	0.67
random data	0.3	16	0.36

Table 2: results of the evaluation

exclude an overfitting of the algorithm to the given data. We give an overview of all results in Table 2. We conclude that target values which reflect in the driver's behavior are predictable with our algorithm.

5 Conclusion and Further Work

With our work we give an opportunity to estimate drivers' properties during driving. This evolutionary algorithm allows optimizing the estimation for different target values. This data could be used for parameterization of assistance systems in cars, so that maybe we can increase the driver's feeling and acceptance of the assistant systems. We conclude that an estimation based on driving behavior is possible.

Yet, to use these results in real world applications, there should be a larger reference database. This could not only increase the estimation quality but it could also allow experiments with other target values.

References

- [1] Bauer, C.; Gonter, M.; Rojas, R.: Fahrerspezifische Analyse des Fahrverhaltens zur Parametrierung aktiver Sicherheitssysteme. In: *Sicherheit durch Fahrassistenz*, Nr. 4. München. 2010.
- [2] Held, P.: *Schätzen verdeckter Fahrereigenschaften auf Basis des Fahrverhaltens*. Master thesis, Otto-von-Guericke-Universität Magdeburg. 2011.
- [3] Winner, H.; Hakuli, S.; Wolf, G.: *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. ATZ-MTZ Fachbuch. Vieweg + Teubner. ISBN 978-3-834-80287-3. 2009.
- [4] Goldhammer, F.; Hartig, J.: Interpretation von Testresultaten und Testeichung. In: *Testtheorie und Fragebogenkonstruktion* (Moosbrugger, H.; Kelava, A., Hg.), Springer-Lehrbuch, S. 165–192. Springer Berlin Heidelberg. ISBN 978-3-540-71635-8. 2008.
- [5] Aamodt, A.; Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun.* 7 (1994), S. 39–59.
- [6] Kruse, R.; Borgelt, C.; Klawonn, F.; Moewes, C.; Ruš, G.; Steinbrecher, M.: *Computational Intelligence*. Vieweg+Teubner Verlag, Wiesbaden, Germany. ISBN 978-3-8348-1275-9. 2011.
- [7] Hastie, T.; Tibshirani, R.; Friedman, J.: *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, 2nd edition Aufl. ISBN 978-0-387-84857-0. 2009.
- [8] Eiben, A.; Aarts, E.; Van Hee, K.: Global convergence of genetic algorithms: A markov chain analysis. In: *Parallel Problem Solving from Nature* (Schwefel, H.-P.; Männer, R., Hg.), Bd. 496 von *Lecture Notes in Computer Science*, S. 3–12. Springer Berlin / Heidelberg. ISBN 978-3-540-54148-6. URL <http://dx.doi.org/10.1007/BFb0029725>. 10.1007/BFb0029725. 1991.
- [9] Gillies, A.: *Machine Learning Procedures for Generating Image Domain Feature Detectors*. Phd thesis, University of Michigan, Michigan. 1985.
- [10] Syswerda, G.: Uniform Crossover in Genetic Algorithms. In: *ICGA*, S. 2–9. 1989.