Dynamic Throttle Estimation by Machine Learning from Professionals

Nathan Spielberg and John Alsterda
Department of Mechanical Engineering, Stanford University

Abstract

To increase the capabilities of an automated Audi TTS, we applied learning algorithms to experimental data from both autonomously and professionally driven test runs to determine transient engine behavior. Data was collected at Thunderhill Raceway by sampling the vehicle's dynamic states over the course of testing. Sparse PCA was then performed to prune the feature set and remove redundant data. Learning transient behavior may increase the control system's performance in tracking a desired velocity profile, compared to the current engine mapping derived under steady state assumptions. To learn this transient behavior, Nonlinear Autoregressive Neural Networks, Multi-layer Perceptron Networks, and Random Forests are used, contrasted, and validated. Due to the highly nonlinear nature of Neural Networks, small perturbations in input states result in physically impossible and undesired predictions; thus, Random Forests prove to be a more robust predictor. Similar unpredictable performance was shown when implementing the multilayer perceptron regressor, which is not included for lack of space. Ultimately, the Random Forest method is chosen to learn the model by estimating the throttle commands required for a given response. The random forest model is verified by predicting race driver data given the current and past vehicle states during testing. Predictions were made with a MSE of 1.8%^{2} and within a 90% confidence bound of ± 2.1%. Additional validation through model inversion provided little insight into performance of the forest, because of correlation with measured velocity and acceleration states. Immediate next steps consist of optimizing the performance of the forest for real time online predictions, and implementation on the vehicle for experimental validation.

Introduction

The Dynamic Design Lab at Stanford University performs vehicle automation research on an Audi TTS: "Shelley." Shelley work focuses on following optimal paths and speed profiles at the limits of friction to achieve lap times competitive with professional racecar drivers. To minimize lap times, the vehicle's controllers must follow speed profiles, as shown in Fig. 1. This requires the controllers' physical models of vehicle dynamics to closely match reality.

Shelley's longitudinal controller operates in a feedback-feedforward framework to follow desired speed. Part of the controller receives a desired longitudinal acceleration and predicts the throttle needed to achieve it. Its current form is a rudimentary lookup table augmented by the engine's gear and RPM state, shown in Fig. 2. This work aims to replace the table with a machine-learned function incorporating many more vehicle and engine states, with delay when appropriate [1].

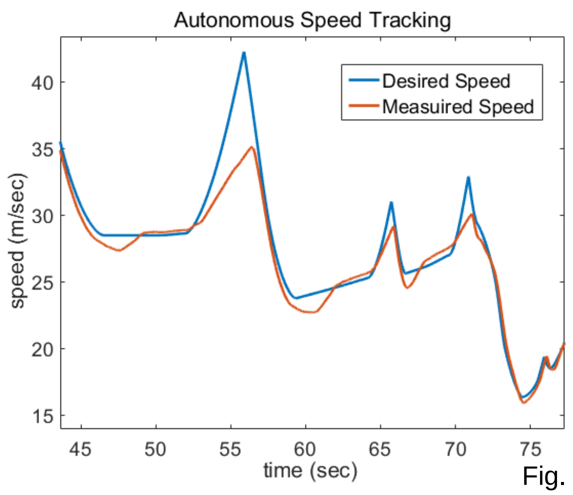


Fig. 1

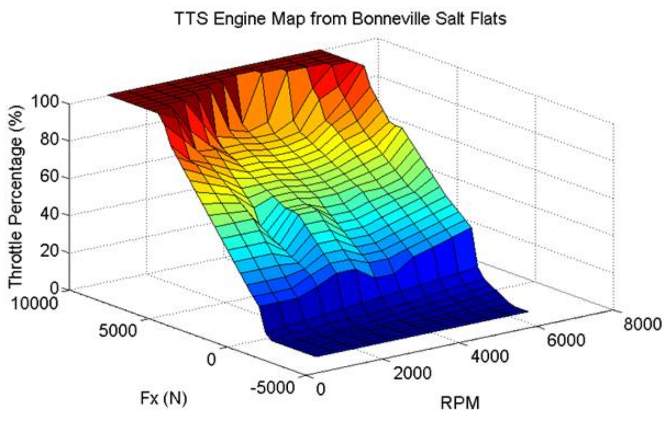


Fig. 2

Problem Representation

The Dynamic Design Lab has recorded several years of Shelley's data sets, consisting of both human and autonomously driven experiments. We selected a small subset of these data: two autonomous and one professionally driven sets to train our models, and one professionally driven set for validation. In total, the sets include 58,278 chronological feature-target pairs, of which 28% form the validation set.

These data consist of vehicle and engine states captured from Shelley's onboard sensors at a rate of 200Hz. The throttle, or pedal percentage actuated at each time step formed our target set, Y. Twelve variables that might intuitively relate to throttle compose our feature set, X. Then, to capture the time dependent dynamics of the system, the feature set was transformed to incorporate delay; each example's feature set was augmented with the features of the five preceding examples to form Φ(X). Finally, an estimation function was learned to predict the throttle paired to each time step's features, shown in Fig. 3.

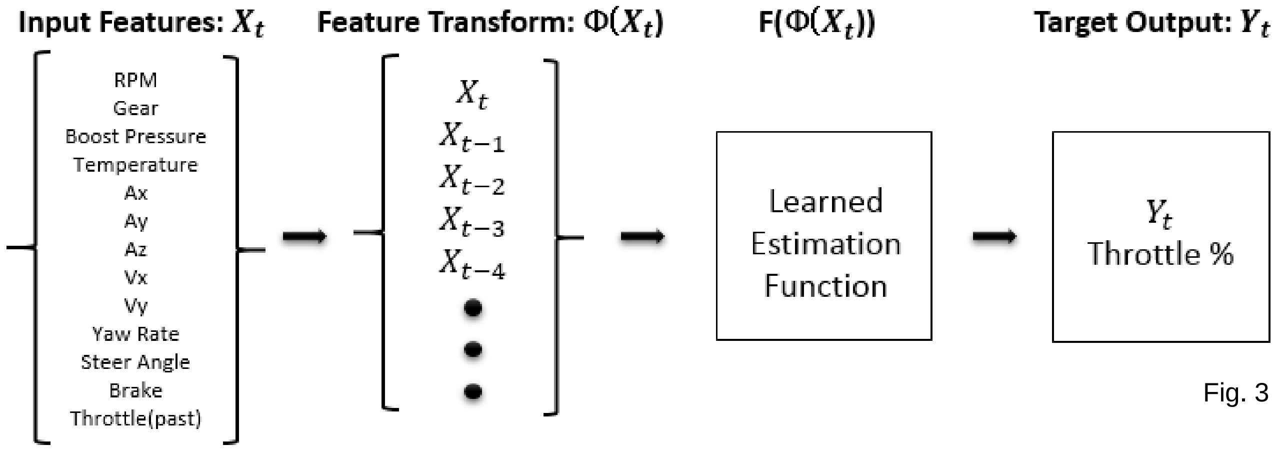


Fig. 3

Additionally, we performed Sparse Principal Component Analysis (SPCA) on our datasets to investigate to the independence and potential pruning of features. Unnecessary features may slow model computation, wasting precious µseconds in the real time environment we hope to embed our model. SPCA is similar to traditional PCA, but modified to limit the number of features incorporated into each PC:

$$X = U\Sigma V^T$$

X = Feature Set Matrix	U = Non-Unitary Basis
Σ = Singular Values	V = Loading Matrix

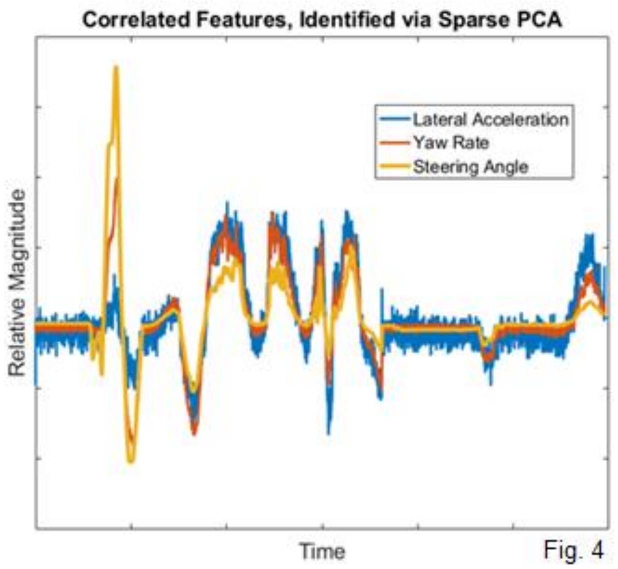
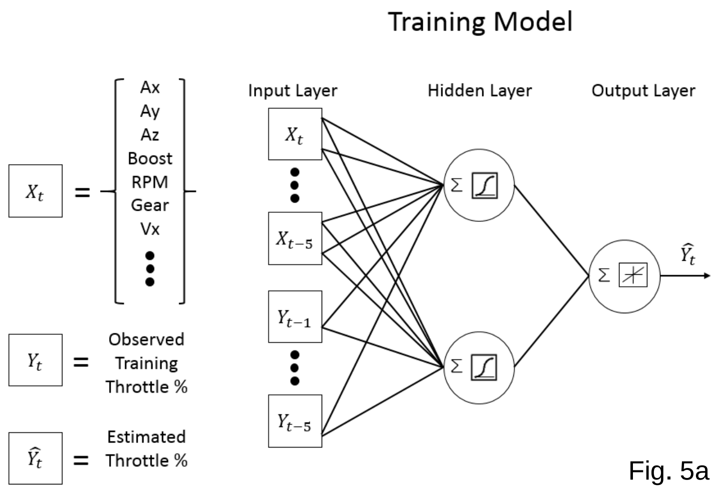


Fig. 4

Traditional PCA is not as helpful to identify feature correlation, because each PC will be typically built from components of every feature. After modification however, SPCA concisely builds each PC from only those features most correlated to one another [7]. We chose the optimal number of features by experimentally increasing the SPCA feature limit. Three was deemed optimal for our data because additional components contributed less than 10% to the loading vectors of ranking PCs. Lateral Acceleration, Yaw Rate, and Steering Angle were found to comprise the highest scoring PC with similar weights; thus these features were suspected to be redundant. A segment of their time series is shown in Fig. 4 which illustrates similarity in signal content. Removing Yaw Rate and Steering Angle from our feature set resulted in a negligible change in performance, and were not included in the methods and results which follow.

Methods / Results

We first attempted to train a neural network to predict throttle percentage. MATLAB's recurrent Nonlinear Autoregressive Neural Network with Exogenous Input (NARX) was chosen due to its design for time-series data. The network is autoregressive because past target values Y are fed into the model as features in parallel to exogenous inputs X. During training, the algorithm is run in open loop, meaning the observed target values are used as past throttle states, shown in Fig. 5a. In validation however, the observed throttle is withheld, and past throttle predictions are input to the model in a closed loop fashion, shown in Fig. 5b. Parameters were updated through Bayesian Regularization Backpropagation (BR), which minimizes a combination of squared errors and weights to produce a network that generalizes well [6]. MATLAB recommended BR over Levenberg-Marquardt and Scaled Conjugate Gradient methods for our noisy data, and experimental validation of performance confirmed this choice [4].


Training Model
X_t = [Ax, Ay, Az, Boost, RPM, Gear, Vx]
Y_t = Observed Training Throttle %
Ŷ_t = Estimated Throttle %
Input Layer
Hidden Layer
Output Layer
X_t
X_{t-5}
Y_{t-1}
Y_{t-5}
Ŷ_t
Fig. 5a

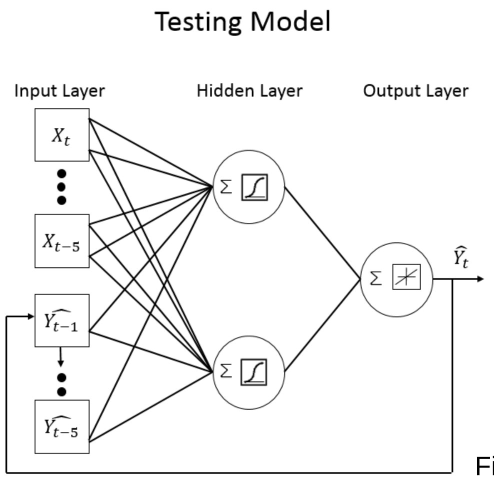


Fig.5b

NARX network design parameters include the number of hidden neurons and number of delay states. In Fig. 6, a learning surface demonstrates that increasing either parameter increased network performance. Conceptually, more neurons may increase the network's complexity, while additional delay states may allow the network to appreciate more time dependent dynamics. Beyond the measurements shown, computation became prohibitively time consuming. 12 hidden neurons and 5 delays states, which minimize the MSE in Fig. 6, were used to build the network responsible for the following results.

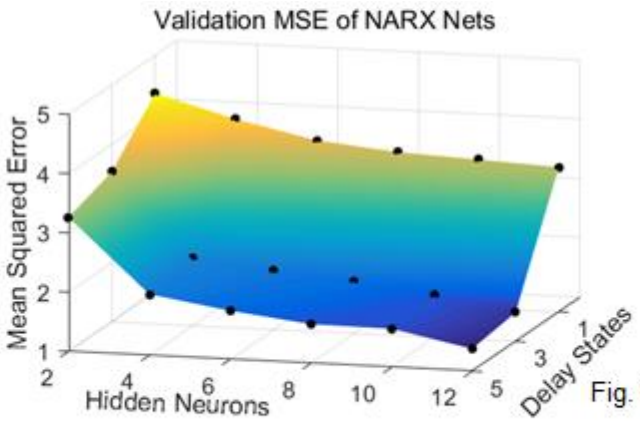


Fig. 6

The NARX network's training and test performance is illustrated in Fig. 7. In training, the network achieved a MSE of 1.5%^{2}. Validation testing was performed using the additional professional driver set. On this new data, results were not consistent, sometimes yielding

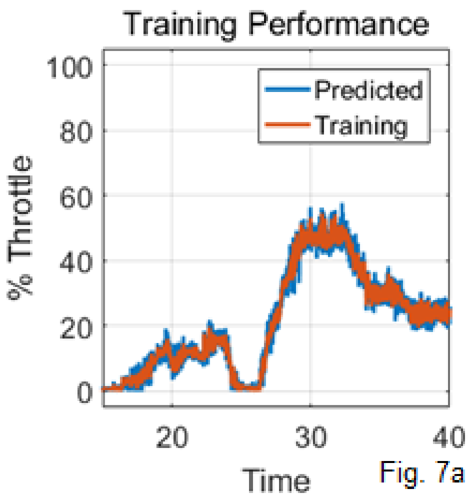


Fig. 7a

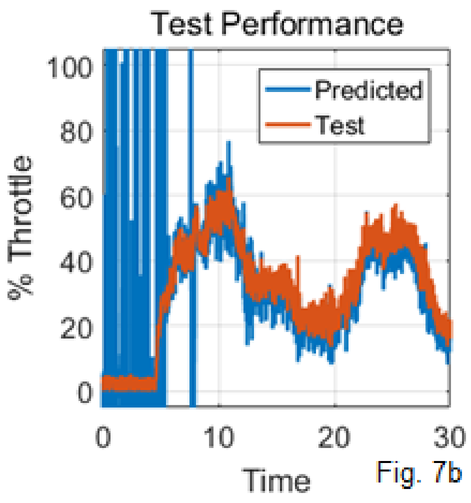


Fig. 7b

physically impossible throttle predictions as shown in Fig. 7b. In the high-liability field of human transportation, the unstable characteristics shown discredit the NARX network; beyond this point our focus narrowed to the Random Forest algorithm.

Using Scikit-Learn, we were also able to apply a Random Forest algorithm to estimate the throttle percentage of the vehicle over the course of our time-series testing data [5]. A random forest is an ensemble method that uses decision trees for regression problems. The randomness of the forest is exhibited by random features that are selected for nodes in decision making (attribute bagging) and that random subsets of the data are used in training each tree (bagging) [2].

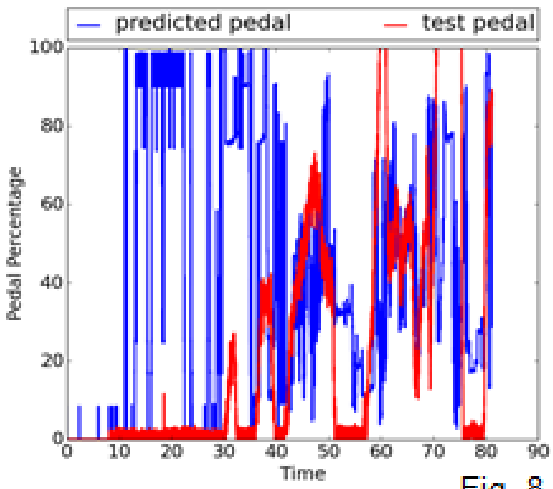


Fig. 8

The forest is constructed using the same training data as the NARX network, consisting of both autonomous and professionally driven tests. To let the forest predict any output value in the training set, we did not limit the number of terminal leaves that each tree could have. In future optimization for computational speed, this feature of the model may be necessary. The feature set comprising the data consists of the vehicle inputs plus the time delayed feature transformation of these inputs as shown in Fig. 3. If the feature set is not time delayed, the MSE of the model reaches over 31%^{2}, as shown in Fig. 8, which was unacceptable. In training the forest, actual past throttle states are used from training data, whereas in testing past throttle values are delayed states of the previous predictions, in a similar closed loop fashion to the NARX network.

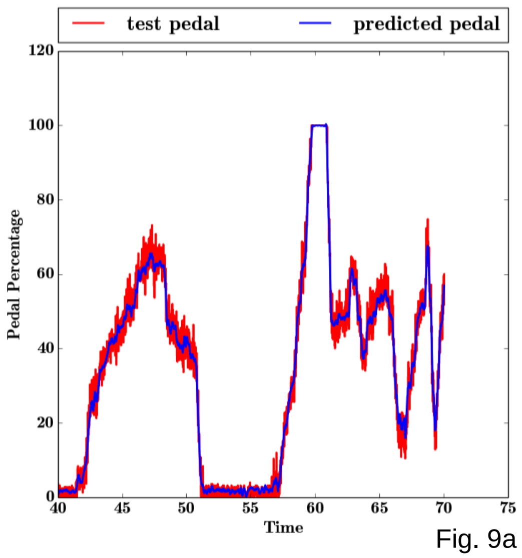


Fig. 9a

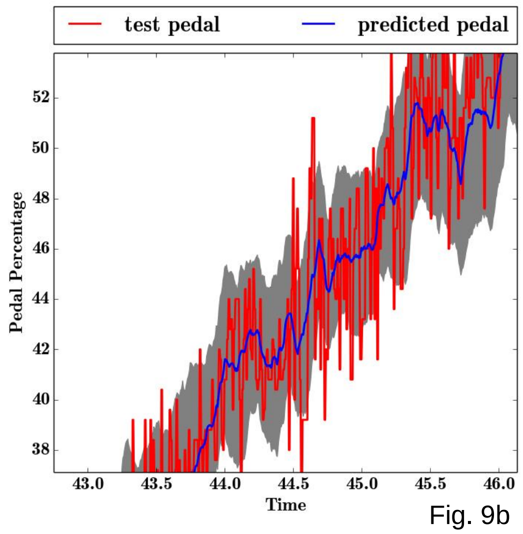


Fig. 9b

Model validation was performed using a separately recorded data set of a professional driver as shown in Fig. 9a. The current and time delayed vehicle states are then used in the closed loop forest model to estimate the driver's throttle percentage over the course of the testing data. Additional validation was attempted using model inversion, by predicting the acceleration during the driver's test using previously predicted throttle percentages. This validation proved unsuccessful, as the error was less than the original forward model. We hypothesize acceleration was predicted primarily from velocity and not previously predicted throttle percentages.

In tuning the forest for best testing performance and lowest generalization error, number of trees were experimentally varied. As shown in Fig. 10, MSE decreased with added trees in the forest, and returns diminished to less than 1% after 80 trees. With increasing number of random trees in the forest, the model can represent more of the bagged testing data as well as attribute bagged parameters, leading to a model with

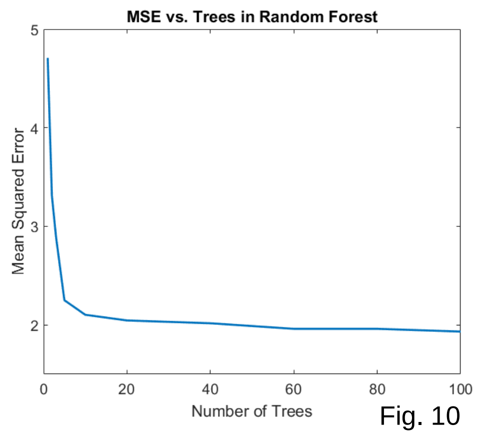


Fig. 10

lower generalization error. Even though MSE decreased with increasing number of trees, increasing number of trees can lead to overfitting in data noise as shown in Fig. 9a and increased computation time in both testing and training [3].

In validating our predictions against professional driver data, using a forest of 80 trees, we estimated the actual throttle percentage with a MSE of 1.8%^{2} with a 90% confidence interval of $\pm 2.1\%$. Confidence intervals were calculated by considering the upper and lower 5% of predictions made by individual trees in the forest for each given prediction as shown in Fig. 9b. An observed 84% of our testing data fell within our 90% interval indicating that our model has some generalization error from testing to training predictions.

Conclusion

This work shows strong potential to provide a throttle function which may outperform Shelley's current empirical map. The Random Forest algorithm, supplied with vehicle and engine states which are available in real time, demonstrates capability to accurately predict the throttle necessary to achieve desired accelerations within a MSE of 1.8%^{2} and within a 90% confidence bound of ± 2.1%. Even though NARX networks were able to attain a MSE of 1.5%^{2}, the computational time required for training and predictions provided limitations on utility. Furthermore, the NARX network showed limited utility because of its propensity to generate undesirable and physically unattainable throttle values. Similar performance was shown when implementing the multilayer perceptron regressor, which is not included for lack of space.

In the near future, we will integrate a pruned random forest model onto Shelley for experimental validation. To do so, the model must be optimized for real time performance and the existing control architecture. Future work in limiting the number of leaves, pruning features, and optimizing for number of trees in the forest will allow us to limit computation time, and thus create a model that we are able to experimentally validate on the car. Further work in investigating the impact of delay states will also allow us to consider adding future states to the feature set. Future desired acceleration states are available in more advanced control frameworks, such as Model Predictive Control (MPC). This may allow our model to learn the causality of future accelerations. Expanding this work in other domains of Shelley's controller such as steering and brake systems may also be able to improve system performance. We have shown the ability to accurately learn and predict the throttle profile for a given test from a professional, with the goal increasing the speed tracking capabilities of the vehicle and minimizing lap times.

Works Cited

[1] Kritayakirana, Krisada, and J. Christian Gerdes. Controlling an Autonomous Racing Vehicle. ASME Dynamic Systems and Control Conference (2009)

[2] L. Breiman. Random forests. Machine Learning, 45(1):5–32, (2001)

[3] M. Segal. “Machine Learning Benchmarks and Random Forest Regression”, Center for Bioinformatics & Molecular Biostatistics,14 Apr. 2003

[4] Neural Network Toolbox™ Reference. Vers. R2016a. Natick, MA: The MathWorks, Inc.

[5] Pedregosa et al. Scikit-learn: Machine Learning in Python. JMLR 12 , 2825-2830, (2011)

[6] Xinxing Pan; Lee, B.; Chunrong Zhang, "A comparison of neural network backpropagation algorithms for electricity load forecasting," Intelligent Energy Systems, 2013 IEEE International Workshop on, pp.22,27, 14-14

[7] Zou, Hui, Trevor Hastie, and Robert Tibshirani. Sparse Principal Component Analysis. Journal of Computational and Graphical Statistics 15(2): 265-286, (2006)