Scalable Robust Learning from Demonstration with Leveraged Deep Neural Networks

Sungjoon Choi, Kyungjae Lee, and Songhwai Oh

***Abstract*— In this paper, we propose a novel algorithm for learning from demonstration, which can learn a policy function robustly from a large number of demonstrations with mixed qualities. While most of the existing approaches assume that demonstrations are collected from skillful experts, the proposed method alleviates such restrictions by estimating the proficiency level of each demonstration using the proposed leverage optimization. Furthermore, a novel leveraged cost function is proposed to represent a policy function using deep neural networks by reformulating the objective function of leveraged Gaussian process regression using the representer theorem. The proposed method is successfully applied to autonomous track driving tasks, where a large number of demonstrations with mixed qualities are given as training data without labels.**

I. Introduction

Traditionally, robots have been mostly deployed in static and structured environments, such as industrial factories, with manually programmed policies. Recently, robots are gradually permeating into our daily living spaces and more complex operations are often required, e.g., autonomous driving [1] or socially adaptive path planning [2]. In such cases, the previous approach of manually describing what a robot should do has a clear limitation.

Consequently, imitation learning has been getting more attention as it can teach a robot not by manually designing a policy or cost function, but by providing demonstrations from human experts. In particular, it has great advantages over traditional methods when it comes to modeling a complex policy function with multiple desiderata to be traded off [3]. While each of the criteria is easy to model, combining them to come up with one reliable and skillful driving strategy is not straightforward. On the contrary, it is more natural to demonstrate driving behaviors and let the agent learn from demonstrations [3], which is also referred to as apprenticeship learning.

While imitation learning is an attractive solution to many complex robot learning problems, it requires a sufficiently large number of proper demonstrations from experts for its performance. In practice, however, collecting flawless demonstrations is often onerous since proficiencies among different demonstrators can vary significantly. Furthermore, directly using a set of demonstrations with mixed qualities may result in a disastrous outcome, e.g., catastrophic car accidents.

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2017R1A2B2006136).

S. Choi, K. Lee, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 151-744, Korea (e-mail: {sungjoon.choi, kyungjae.lee}@cpslab.snu.ac.kr, songhwai@snu.ac.kr).

In this paper, we present a scalable method for robust learning from demonstration using leveraged deep neural networks. While leveraged GPs can incorporate additional information about training data, called leverage parameters [4], a conventional learning process of neural networks, e.g., mini-batch learning with a gradient descent update rule, cannot be directly applied to handle leverage parameters. To overcome this issue, we first interpret the objective function of leveraged GPs using the representer theorem [5] and propose a new leveraged cost function, which can be used to train a deep neural network. A deep neural network (DNN) trained with a leveraged cost function will be referred to as leveraged DNNs. Furthermore, leverage optimization is performed in a divide-and-conquer manner with a coordinate ascent method.

The proposed method is applied to autonomous driving tasks where demonstrations are collected from three different driving modes (safe, inexperienced, and aggressive) to verify the capability of the proposed leverage optimization method.[1] We first show that the proposed leveraged optimization method with a coordinate descent method works surprisingly well for discriminating the source of demonstrations. Furthermore, the leveraged DNN trained with the leveraged cost function performs comparable to leveraged GPs for problems with low-dimensional feature representation (simple problem) and outperforms leveraged GPs significantly for problems with high-dimensional feature representation (complex problem) as it can incorporate more demonstrations.

II. Preliminary

A. Leveraged Gaussian Process Regression

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [6]. A Gaussian process defines a distribution over infinite dimensional vectors, i.e., functions, and is completely specified by its mean function $m(x)$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ as follows:

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}),\ k(\mathbf{x}, \mathbf{x}')).$$

If the observation model is $y_i = f(\mathbf{x}_i) + w_i$ with $w_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_w^2)$, the conditional distribution of an output $y_\star$ at an unseen input $\mathbf{x}_\star$ given the training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ becomes

$$y_\star | D \sim \mathcal{N}\left(\mu_\star(\mathbf{x}_\star|D),\ \sigma_\star^2(x_\star|D)\right),$$

[1] While three distinct driving modes are used in the paper for illustration purpose, the proposed framework can handle more realistic driving demonstrations with diverse types of driving modes.

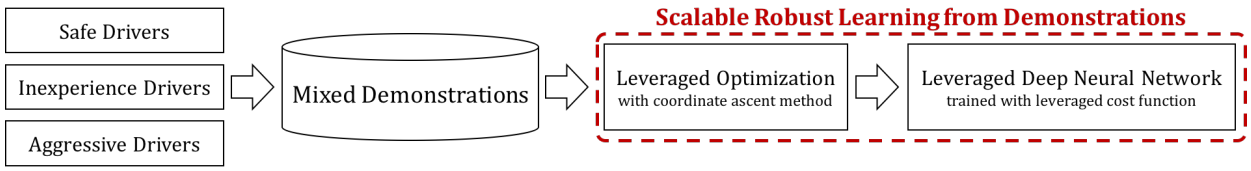

Fig. 1: A flowchart of the proposed scalable robust learning from demonstration method.

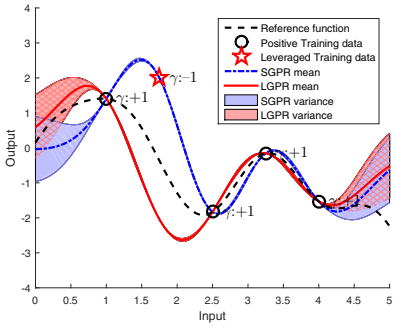

Fig. 2: Regression results from stationary GPR (SGPR) and leveraged GRP (LGPR). The reference function and 4 sampled training data are shown as a black dashed line and circles, respectively. The training sample whose leverage is -1 is shown as a red star. The mean functions for stationary GPR and leveraged GPR are shown as a blue dash line and a red solid line, respectively.

where

$$\mu_\star(\mathbf{x}_\star|D) = k(\mathbf{x}_\star)^T(k(\mathbf{X},\mathbf{X}) + \sigma_w^2 I)^{-1}\mathbf{y} \quad (1)$$

and

$$\sigma_\star^2(\mathbf{x}_\star|D) = k_\star - k(\mathbf{x}_\star)^T(k(\mathbf{X},\mathbf{X}) + \sigma_w^2 I)^{-1}k(\mathbf{x}_\star). \quad (2)$$

Here, $k_\star = k(\mathbf{x}_\star,\mathbf{x}_\star)$, $k(\mathbf{x}_\star) \in \mathbb{R}^n$ is a covariance vector between n data points $\mathbf{X} = [\mathbf{x}_1 \ \dots \ \mathbf{x}_n]$ and the test point $\mathbf{x}_\star$, and $k(\mathbf{X},\mathbf{X}) \in \mathbb{R}^{n\times n}$ is a covariance matrix for $\mathbf{X}$, such that $[k(\mathbf{X},\mathbf{X})]_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$.

A leveraged Gaussian process was proposed in [4] to incorporate not only positive training samples but also negative training samples under a unified regression framework. Figure 2 illustrates the regression results of the ordinary Gaussian process regression (GPR) with a squared exponential kernel function and leveraged GPR given five training data. The regression result of leveraged GPR, shown in the red line, tends to *drift away* from the second training data from the left, which works as a negative training example due to its leverage value of -1, while *anchor* to the rest of positive training data.

To achieve this, a leveraged kernel function was proposed in [4], [7] by introducing a leverage parameter γ at each training sample, which varies from -1 to $+1$:

$$k_{lev}(\mathbf{x}_i,\mathbf{x}_j) = \cos\left(\frac{\pi}{2}(\gamma_i - \gamma_j)\right) k_{\text{PSD}}(\mathbf{x}_i,\mathbf{x}_j), \quad (3)$$

where $k_{\text{PSD}}(\mathbf{x}_i,\mathbf{x}_j)$ can be any positive semidefinite kernel function, and γ_i and γ_j are leverage parameters of the i-th and j-th inputs, respectively. It is assumed that $\gamma_i \in [-1,1]$ for all i. In this paper, we use a popular squared exponential (SE) kernel function as $k_{\text{PSD}}(\cdot,\cdot)$:

$$k_{SE}(\mathbf{x}_i,\mathbf{x}_j) = g^2 \exp\left(-\frac{1}{2}\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{l^2}\right), \quad (4)$$

where $\theta = \{g^2, l^2\}$ is a set of hyperparameters

For test (unseen) inputs, γ can be set to one. This leverage parameter γ has an important meaning in that if γ is -1, the corresponding training sample works as a *negative* training sample while when γ is $+1$ the corresponding training sample works identical to the ordinary *positive* training sample.

III. Scalable Robust Learning From Demonstration

In this section, we present a scalable robust learning from demonstration algorithm using leverage optimization and leveraged DNNs. To learn from demonstrations of mixed qualities, we first optimize the leverage parameter of each demonstration, which will be referred to as leverage optimization. The overall procedure of the proposed method for scalable robust learning from demonstration is shown in Figure 1.

A. Scalable Leverage Optimization

To achieve scalability for leverage optimization with large scale demonstrations, a simple divide-and-conquer method is utilized. Given a set of demonstrations of input and output pairs, we first partition the total demonstrations into subsets of a moderate size and then perform leverage optimization independently for each subset. Furthermore, instead of gradient-based proximal linearized minimization, we used a coordinate ascent method which has been widely used for solving sparse constrained problems [8]. In each iteration, we first run a line search on every leverage parameter and update one leverage parameter which achieves the largest log marginal likelihood. The optimization process stops if there is no improvement in the log marginal likelihood. We have observed that the coordinate ascent method with this simple stopping rule converges to a better solution while gradient based methods often converge to poor local optima.

B. Leveraged Deep Neural Network

Deep neural networks (DNNs), such as multi-layer perceptrons (MLPs) or convolutional neural networks (CNNs), have been widely used in robotics to model complex nonlinear functions [9] due to its scalability and expressiveness. In order to train a DNN, a dataset of input and output pairs and a differentiable cost function is required. In our case, however, a demonstration dataset consists of tuples of an

input, output, and leverage parameter. Therefore, the ordinary learning framework for DNNs cannot be applied directly. In this section, we first interpret the objective function of leveraged Gaussian process regression and obtain a leveraged cost function suitable for training DNNs.

1) Interpretation of Leveraged GPR: Gaussian process regression (GPR) is a nonparametric regression method in which the predictor does not take a predefined form but is constructed using the information derived from the data. As shown in Figure 2, stationary GPR predicts an output value of an unseen location by interpolating given data points.

While (1) is derived from the mean of the Gaussian process posterior distribution, one can also derive it from the representer theorem [5] by minimizing following functional [6]:

$$\begin{aligned} J_{SE}[f] &= \frac{1}{2\sigma_n^2}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 + \frac{1}{2}||f||_{\mathcal{H}}^2 \\ &=: E_{SE}(f|D) + R_{SE}(f), \end{aligned} \tag{5}$$

where $D = \{(\mathbf{x}, y)_i | i = 1, ..., n\}$ is the training data, $||\cdot||_{\mathcal{H}}$ is a Hilbert norm, σ_n^2 is the variance of the measurement noise, and $f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, \mathbf{x}_i)$ is a regressor in the reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ with regressor parameters α_i. We assume that we are using a SE kernel (4) for Gaussian process regression while the use of a different kernel function will not affect the interpretation. The first and second term in (5) work as a data fitting term and a regularizer, respectively. From (5), we can see that the objective of Gaussian process regression is to find a function $f(\cdot)$ in the RKHS which best fits given training data D, i.e., minimizing $E_{SE}(f|D)$, while regularizing the smoothness of the function using the constraint $R_{SE}(f)$.

The leveraged kernel function (3) can be seen as a kernel function of four arguments, i.e., $\mathbf{x}$, γ, $\mathbf{x}'$, and γ'. If we assume $\mathbf{x}$ and γ are elements in $\mathcal{X}$ and $[-1, 1]$, respectively, then the kernel function maps the product space of $\mathcal{X}$ and $[-1, 1]$ to a real line, i.e., $k : (\mathcal{X} \times [-1, 1]) \times (\mathcal{X} \times [-1, 1]) \to \mathbb{R}$.

Applying (3) to the mean function of a Gaussian process, $f(\cdot)$, it becomes

$$\begin{aligned} f(\mathbf{x}, \gamma) &= \sum_{j=1}^{n} \alpha_j k(\mathbf{x}, \gamma, \mathbf{x}_j, \gamma_j) \\ &= \sum_{j=1}^{n} \alpha_j k_\gamma(\gamma, \gamma_j) k_{SE}(\mathbf{x}, \mathbf{x}_j). \end{aligned} \tag{6}$$

If we assume that the training data are either fully positive or negative, which was the case in [4], γ can only have $+1$ or -1, i.e., $\gamma \in \{-1, 1\}$. Then, we can rewrite (5) by substituting (6) as follows

$$\begin{aligned} J_L[f] =& \frac{1}{2}||f||_{\mathcal{H}}^2 \\ &+ \frac{1}{2\sigma_n^2}\sum_{i=1}^{n} \\ &\left(y_i - \sum_{j=1}^{n} \alpha_j \cos\left(\frac{\pi}{2}(\gamma_i - \gamma_j)\right) k_{SE}(\mathbf{x}_i, \mathbf{x}_j)\right)^2 \\ =& \frac{1}{2}||f||_{\mathcal{H}}^2 + \frac{1}{2\sigma_n^2}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{n}\alpha_j\gamma_i\gamma_j k_{SE}(\mathbf{x}_i, \mathbf{x}_j)\right)^2 \\ =:& R_L(f) + E_L(f|D). \end{aligned} \tag{7}$$

If we multiply $\gamma_i^2 = 1$ to the data fitting term $E_L(f|D)$ of (7), we get

$$E_L(f|D) = \frac{1}{2\sigma_n^2}\sum_{i=1}^{n}\left(\gamma_i y_i - \sum_{j=1}^{n}\gamma_j\alpha_j k_{SE}(\mathbf{x}_i, \mathbf{x}_j)\right)^2. \tag{8}$$

$E_L(f|D)$ is now identical to $E_{SE}(f|D)$ with the squared exponential (SE) kernel with inputs $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ and outputs $\{\gamma_1 y_1, ..., \gamma_n y_n\}$. Therefore, the resulting regressor which minimizes (8) will make regression with flipping the sign of the outputs with $\gamma = -1$. For a zero-mean Gaussian process, flipping the sign of outputs of negative training data will make the regressor be far from the negative data.

2) Leveraged Cost Function: An examination of (8) tells us that if all the training data are either fully positive or negative, the output of the leveraged Gaussian process regression will be identical to flipping the sign of the output of the negative training data whose leverages are -1 and we can use the original Gaussian process regression approach. In other words, we can combine the leverage parameters, γ, and output training data, y, to come up with a new training dataset, which consists of new input and output pairs, i.e., $\{\mathbf{x}_i, \gamma_i y_i\}_{i=1}^n$, and use them to train a deep neural network with conventional squared loss function, i.e.,

$$\mathcal{L}_\theta = \frac{1}{n}\sum_{i=1}^{n}(f_\theta(\mathbf{x}_i) - \gamma_i y_i)^2 + R(\theta), \tag{9}$$

where $f_\theta(\cdot)$ is a deep neural network and $R(\theta)$ is a regularizer, e.g., l_2 weight decay.

However, there is one remaining problem in that the result in Section III-B.1 is based on the assumption that all the training data are either fully positive or negative, i.e., having only $+1$ or -1 leverage values. However, the optimized leverage parameters can have values between -1 and $+1$. If we recall the property of leveraged Gaussian processes, the training data whose leveraged is 0 has no effect on the resulting regressor. To incorporate this property, we modified (9) as follows:

$$\mathcal{L}_\theta = \frac{1}{n}\sum_{i=1}^{n}|\gamma_i|(f_\theta(\mathbf{x}_i) - \gamma_i y_i)^2 + R(\theta) \tag{10}$$

where $|\gamma_i|$ is inserted inside the summation to provide less weight to a sample with a small leverage value in magnitude. A deep neural network trained with (10) will be referred to as a leveraged DNN.

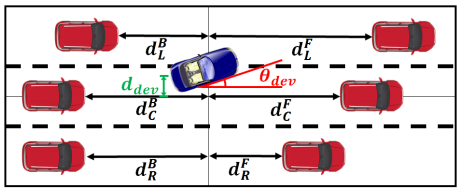

Fig. 3: Track driving simulator and feature descriptions.

IV. Autonomous Driving Experiments

In this section, we focus on the scenario of autonomous track driving experiments. Existing learning from demonstration methods assume that demonstrations are collected from skillful experts [10], or at least some additional information to assess the quality [4]. In this experiment, however, we assume that demonstrations are collected from multiple demonstrators with mixed proficiencies.

Particularly, driving demonstrations are collected from three different driving styles: safe driver, inexperienced driver, and aggressive driver. Both safe and inexperienced drivers avoid collision with other cars in front whereas the aggressive driver tries to hit the car in front. While the safe driver maintains her lane, the inexperienced driver tends to stick to the right side of the lane.[2]

We define a *driving policy* π as a mapping from input features to the trigonometric encoded desired heading angle of a car, $(\cos\theta_{dev}, \sin\theta_{dev})$. Figure 3 illustrates obtainable features of the track driving simulator and two different feature representations are used throughout this section. One is a four-dimensional frontal feature representation, which consists of three frontal distances to the closest cars in left, center, and right lanes and lane deviate distance, $(d_L^F, d_C^F, d_R^F, d_{dev})$, and the other is a seven-dimensional frontal and rearward feature representation, which consists of three frontal distances to the closest cars in left, center, and right lanes in the front, three rearward distances to the closest cars in left, center, and right lanes in the back, and lane deviate distance, $(d_L^F, d_C^F, d_R^F, d_L^B, d_C^B, d_R^B, d_{dev})$.

Once a driving policy is trained, control of a car is done by a simple feedback controller by changing the angular velocity to minimize the error between current heading of a car and the output of the policy function. A directional velocity is fixed for 72 km/h and the control frequency is set to 10 Hz. While we use a simple unicycle dynamic model, more complex dynamic models, e.g., vehicle and bicycle dynamic models, can be also used.

In order to collect a sufficient number of driving demonstrations, we first collect driving demonstrations by manually controlling the car in a simulated environment with changing the locations of other cars. Then, density matching reward learning [11] is used for generating diverse driving demonstrations by first learning a driving policy of each mode with manually collected demonstrations and collecting a sufficient number of demonstrations autonomously by changing the track settings. The demonstrations of three different modes are combined without knowing which mode they are collected from. Figure 4 shows three different driving policies (desirable heading at each position) with black arrows and controlled trajectories of each mode with blue plain lines and red dotted lines indicating safe and collided trajectories, respectively.

[2]This inexperienced driving behavior is motivated by novice drivers who try to put the driver himself to the center of the lane making his car stick to the right where the steering wheel is assumed to on the left.

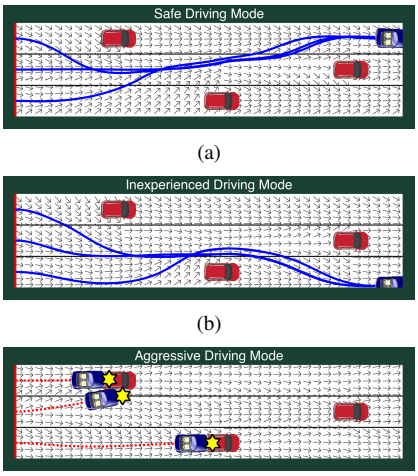

Fig. 4: Resulting policy flow and trajectories of three different driving modes: (a) safe driving mode, (b) inexperienced driving mode, and (c) aggressive driving mode.

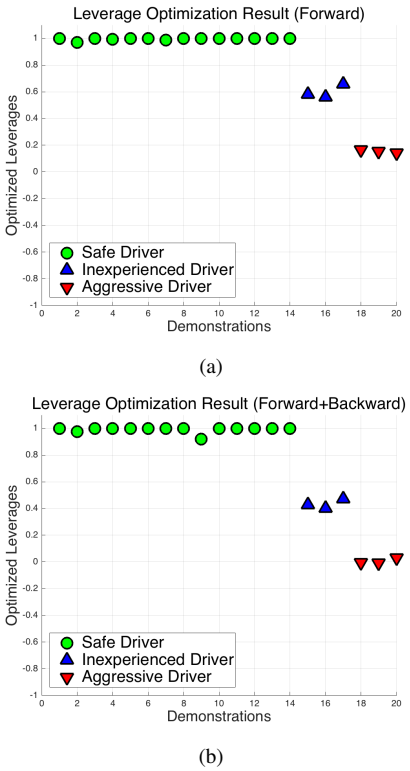

Fig. 5: Leverage optimization results on two different feature representations: four-dimensional frontal feature representation (Forward) and seven-dimensional frontal and rearward feature representation (Forward+Backward).

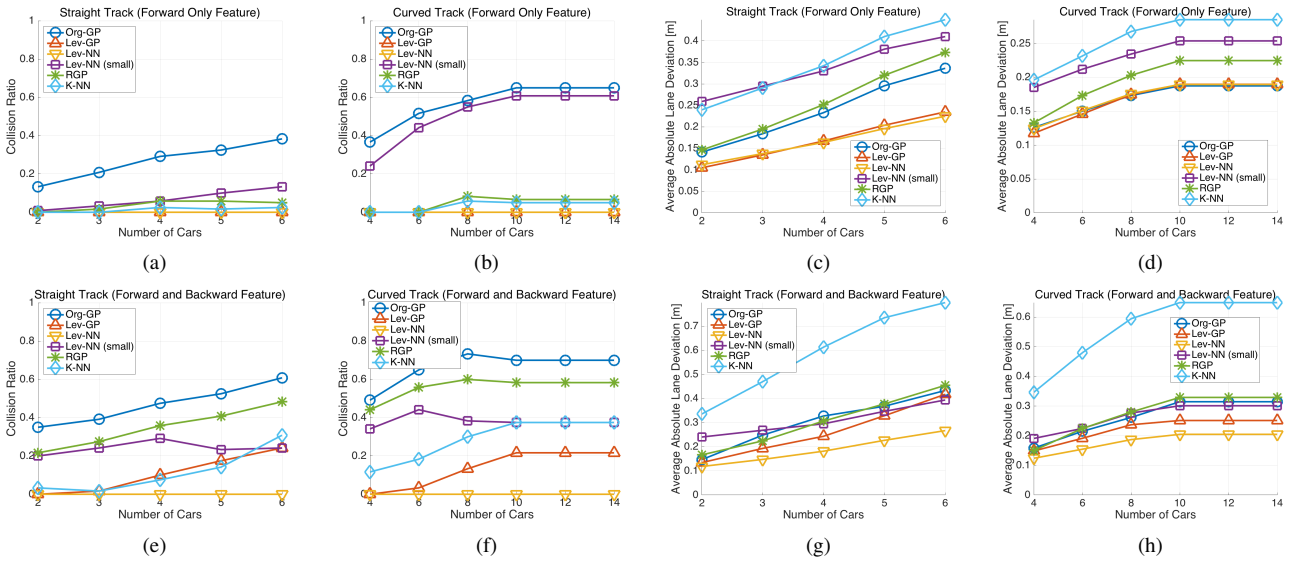

Fig. 6: Average collision ratio and average absolute lane deviation distances of tested four algorithms on straight and curved tracks with two different input feature representations.

A. *Leverage Optimization Results*

To optimize leverage values of a large number of demonstrations, a simple divide-and-conquer approach is used. In particular, each partition has 20 demonstrators consists of 14 safe drivers, 3, inexperienced drivers, and 3 aggressive drivers, where each driver generates 100 demonstrations. Each demonstrator has a leverage value of her own. We collected 20 independent sets of $2,000$ demonstrations with a fixed driving style ratio and perform the leverage optimization with a coordinate ascent method on each mini-batch.

Leverage optimization results on a set of $2,000$ demonstrations are shown in Figure 5, where two different feature representations are shown. In both cases, the proposed leverage optimization method correctly identifies the source of demonstrations by assigning similar leverages. Furthermore, the leverage optimization assigns high leverages $(+1)$ to safe drivers and low leverages to inexperienced drivers (0.5) and aggressive drivers (0). The optimized leverages are further being used to train both leveraged Gaussian processes and leveraged DNNs with the proposed leveraged cost function (10).

B. *Autonomous Driving with Leveraged Deep Neural Networks*

Once the leverage values of demonstrations are optimized with a divide-and-conquer method, four different policy functions are trained for two different input feature representations. The first two policies are trained with a Gaussian process (GP) using $4,000$ demonstrations with and without the leverage optimization results, which will be referred to as original GP and leveraged GP. $4,000$ demonstrations are collected from two sets of $2,000$ demonstrations in Section III-A. Other polices are trained using the proposed leveraged DNNs trained with the leverage optimization results from a total of $20,000$ and $4,000$ demonstrations, which will be referred to as leveraged DNN and leveraged DNN-small, respectively. In particular, leveraged GP and leveraged DNN-small shares the same training set. When training a leveraged DNN, the leveraged loss function (10) is used.

For the frontal feature representation, the neural network has two hidden layers with 256 hidden units per layer with a hyperbolic tangent activation function. For the frontal and rearward feature representation, the neural network has two hidden layers with 512 hidden units with the same activation function. In all cases, Adam optimizer [12] with 0.01 learning rate is used for $1,000$ epochs. We use TensorFlow [13] to optimize the weights of leveraged DNNs.

We evaluated the performances of six different algorithms, original GP, leveraged GP, leveraged DNN, leveraged DNN-small, LfD with k-nearest neighborhood (KNN) [14], and robust GP (RGP) [15] on two different track environments: straight and curved tracks in Figure 7, while varying the number of deployed cars. All cars are assumed to be static. We quantitatively measure the performance in terms of two different criteria: the average collision ratio and average absolute lane deviation distance. Both criteria are averaged over 30 independent runs by changing locations of other cars and the initial lane of the controlled car.

The first and second rows of Figure 6 show average collision rates and average absolute lane deviation distances with frontal features and frontal and rearward features, respectively. When using the frontal features (first row), policy functions with both leveraged GP and a leveraged DNN show no collision with other cars in both straight and curved track whereas the collision ratio of the original GP increases as the number of cars increases. We would like to note that the leveraged DNN trained with the same demonstrations shows poor performance compared to the leveraged GP. We believe this indicates that given the same number of training data, the generalization performance of GPs is better than that of DNNs for this case.

However, when using frontal and reward features, the

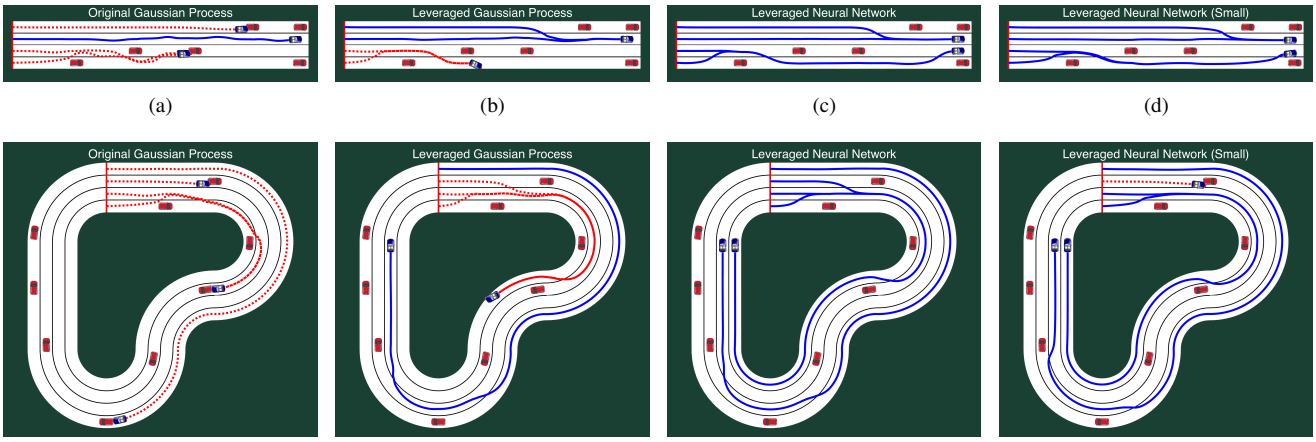

Fig. 7: Controlled trajectories in straight tracks.

proposed leveraged DNN outperforms the leveraged GP in terms of both collision ratio and absolute lane deviation distance. This is mainly because that the leveraged DNN is trained with $20,000$ demonstrations while the leveraged GP is trained with only $4,000$ demonstrations.[3]

The controlled trajectories of tested algorithms on straight and curved tracks are shown in Figure 7, respectively. The first and second row show resulting trajectories using four-dimensional frontal features and seven-dimensional frontal and rearward features, respectively. In both straight and curved tracks, a leveraged DNN trained with $20,000$ demonstrations show comparable performance against leveraged GPR when the frontal feature representation is used, and outperforms all other methods for the case with the frontal and rearward feature representation in terms of both collision ratio and average absolute lane deviations.

V. Conclusion

In this paper, a scalable learning from demonstration method, which can robustly learn a policy function from demonstrations with mixed qualities, is presented. In particular, a leveraged cost function is proposed by interpreting the objective function of leveraged GPR using the representer theorem and further used to train a leveraged DNN. The proposed scalable robust learning from demonstration method is applied to autonomous driving experiments where driving demonstrations are collected from three different driving modes, safe driving mode, inexperienced driving mode, and aggressive driving mode. Leveraged DNNs perform comparably to leveraged GPR for simple problems but outperforms leveraged GPR for more complex problems as it can incorporate more demonstrations, demonstrating its scalability.

VI. Acknowledgment

The authors would like to thank Dong-Hyun Lee for valuable comments and suggestions on training deep neural networks.

[3] 4,000 was the largest number of demonstrations we could use to train leveraged GPR.

References

[1] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[2] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning," *International Journal of Social Robotics*, vol. 8, no. 1, pp. 51–66, January 2015.

[3] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. of the 21st International Conference on Machine learning*, July 2004.

[4] S. Choi, E. Kim, K. Lee, and S. Oh, "Leveraged non-stationary Gaussian process regression for autonomous robot navigation," in *Proc. of the International Conference of Robotics and Automation*. IEEE, 2015.

[5] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Computational learning theory*. Springer, 2001, pp. 416–426.

[6] C. Rasmussen and C. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 1.

[7] S. Choi, K. Lee, and S. Oh, "Robust learning from demonstration using leveraged Gaussian processes and sparse constrained opimization," in *Proc. of the International Conference of Robotics and Automation*. IEEE, May 2016.

[8] T. T. Wu and K. Lange, "Coordinate descent algorithms for lasso penalized regression," *The Annals of Applied Statistics*, pp. 224–244, 2008.

[9] S. Levine and V. Koltun, "Guided policy search," in *Proc. of the International Conference of Machine Learning (ICML)*, 2013.

[10] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[11] S. Choi, K. Lee, A. Park, and S. Oh, "Density matching reward learning," *arXiv preprint arXiv:1608.03694*, 2016.

[12] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[14] D. C. Bentivegna and C. G. Atkeson, "Learning from observation using primitives," in *Proc. of the International Conference of Robotics and Automation*. IEEE, 2001.

[15] E. Kim, S. Choi, and S. Oh, "A robust autoregressive Gaussian process motion model using l1-norm based low-rank kernel matrix approximation," in *Proc. of the International Conference of Intelligent Robots and Systems (IROS)*. IEEE, 2014.