Clustered Trajectory Motion Prediction for Human-Robot Collaboration

Alex Cuellar
Interactive Robotics Group.
MIT
Cambridge, MA
alexcuel@mit.edu

NOTE: This writeup is not intended to be a finished project or ready for publication. It is an in-depth snapshot of a UROP project under Prof. Shah thus far.

***Abstract*—As automation becomes more integral to workplaces, homes, and other spaces occupied by humans, robots require the ability to predict and avoid humans. In this project, I propose an algorithm able to predict the trajectory of a robot's human participant through space given observed clustered trajectories. Given that each human and environment will present unique sets of trajectories, I focused on developing an algorithm able to learn online and make predictions with limited data.**

I. Introduction

While robot motion planning as a field has seen significant of research in past years, the presence of humans requires any planning system to react in a changing environment. Therefore, a robot must have some ability to predict the the human's position over time to effectively plan. Online alignment and clustering algorithms such as BEST-PTA [1] tackle part of this problem by grouping similar segments of a human's path. These clusters can give some indication of a person's position during one piece of a segmented trajectory. However, these methods do not use information from an incomplete trajectory to improve predictions of the human's trajectory once assigned to a cluster. In this paper, we introduce an algorithm which makes informed predictions of a human's partial trajectory for one segmented action given previously clustered data from BEST-PTA.

To more formally define the problem, let a trajectory be represented by $X \in \mathbb{R}^{N \times T}$, where N is the number of dimensions over which we are tracking (i.e. 3 when using spatial trajectories) and T is the number of time steps. Also denote $\vec{x}_t$ as the coordinates of trajectory X at time step t. Additionally, let X^P be the partial trajectory for an action where we have observed the first l time steps of the action. Lastly, denote the set of all completely observed trajectories of an action as X^O, where the j-th individual trajectory in this set is denoted X^O_j. Our goal is to predict the time steps $t > l$ of X^P using the previously observed trajectories X^O.

II. Related Works

Many methods to predict human motion have been presented for various applications and with various constraints. [2] trains a GHMM to discretize an environment as a graph of important spacial points through which movement can be modeled and predicted. However, The main goal of this research focused more on estimating the endpoint of an action or trajectory segment. While the algorithm could extrapolate likely nodes in the graph across which the human would likely move, the result gives little granularity to any continuous (or at least small time step) change in position.

Many others use deep learning of some kind to make predictions. [3], [4], and [5] all use LSTMs to predict paths forward. None assume clustering of prior paths or well-defined start and end-points. Additionally, they do take into account social forces by others pedestrians present. However, while [3] and [5] looks at features relating to absolute locations of people, [4] utilizes the relative distances between pedestrians. All of these also make predictions from images, not directly form a person's spacial position.

[6] proposes a prediction algorithm for vehicles, especially during turns. This action-specific prediction is the closest to our proposed goal. It also makes use of Gaussian Mixture Models instead of more data-hungry methods from deep learning. However, it uses Gaussian Mixture Models to predict the coefficients of a Chebyshev polynomial representation of a path instead of acting on points themselves. Therefore, the complexity of any predicted path is limited by the order polynomial predicted by the algorithm.

III. Method

To solve the problem of clustered human motion prediction, we use iterative instances of Gaussian Process Regression to predict the change in a partial trajectory's position from one time step to another. Section $3A$ explains the high-level framework of our iterative prediction. $3B$ describes the formulation of each GPR iteration. $3C$ describes our choices of kernel functions for the GPR. $3D$ describes methods for hyperparameter optimization for GPR. Lastly $3E$ describes a complimentary filter applied to a predicted trajectory to produce more reliable results.

A. Iterative Framework

Our algorithm is initially given a partial trajectory whose last observed point is $\vec{x}^P_l$. First, we find the KNN points of $\vec{x}^P_l$ with respect to the 3 spatial dimensions and time, where time is defined as number of time steps since the start of

the trajectory (e.g. the four dimensional vector $(\vec{x_l^P}, l)$). These neighbors are used as input to a GPR which predicts the spatial change between $\vec{x}_l^P$ and the first unknown time step. Denote this change as $\Delta\hat{\vec{x}}_{l+1} = (\Delta\hat{x}_{x,l+1}, \Delta\hat{x}_{y,l+1}, \Delta\hat{x}_{z,l+1})$. Therefore, the first point of our prediction will be:

$$\hat{\vec{x}}_{l+1}^P = \vec{x}_l^P + \Delta\hat{\vec{x}}_{l+1} \tag{1}$$

For the next iteration, we find the KNN of $\vec{x}_{l+1}^P$ from X^O to inform another instance of GPR and predict $\vec{x}_{l+2}^P$. We continue this iterative process until we make enough predictions such that the combined lengths of X^P and the predicted trajectory reach mean length of trajectories from X^O.

B. GPR Formulation

When we find the nearest neighbors to a point of interest $\hat{\vec{x}}_i^P$, we use them as the data for a GPR with 7 features. 3 features are the (x, y, z) coordinates of a neighbor, one is the number of time steps after the start of its respective trajectory, and the final 3 features are defined as the direction in which the trajectory is moving. This direction for a point $\vec{x}_{j,t}^O$ is defined as:

$$\Delta\vec{x}_{j,t}^O = \vec{x}_{j,t}^O - \vec{x}_{j,t-1}^O \tag{2}$$

Denote the 7-dimesnonal feature vector of the j-th nearest neighbor as $\vec{a}_j^O = (\vec{x}_{i,t}^O, t, \Delta\vec{x}_{i,t}^O)$. We extract the same features from $\hat{\vec{x}}_i^P$. Here, the direction features are based on:

$$\Delta\hat{\vec{x}}_{l+i}^P = \begin{cases} \hat{\vec{x}}_i^P - \hat{\vec{x}}_{i-1}^P & i > l+1 \\ \hat{\vec{x}}_{l+1}^P - \vec{x}_l^P & i = l+1 \\ \vec{x}_l^P - \vec{x}_{l-1}^P & i = l \end{cases} \tag{3}$$

and call this feature vector $\vec{a}^P = (\hat{\vec{x}}_i^P, i, \Delta\hat{\vec{x}}_i^P)$. With these features, we define covariance matrices to compare the similarity between pairs of points in the neighbors:

$$C_{OO}(\vec{a}_j^O, \vec{a}_k^O) = f(\vec{a}_j^O, \vec{a}_k^O) \tag{4}$$

and between the neighbors and the feature vector $\vec{x}_i^f$:

$$C_{OP}(\vec{a}_j) = f(\vec{a}_j^O, \vec{a}^P) \tag{5}$$

where f is a kernel function. The choice of kernel function is explained in section 3C. We use these covariance matrices to infer the spatial change of our predicted trajectory along one dimension. Let $\vec{y}_{i,x}$, $\vec{y}_{i,y}$, and $\vec{y}_{i,z}$ be the vectors containing the x, y, and z portions of $\Delta\vec{x}_{i,t+1}^O$ across every point in our nearest neighbors.

Therefore, GPR will predict each dimension of $\Delta\hat{\vec{x}}_{i+1}^P$ individually. For example, along the x dimension, the change is:

$$\Delta\hat{x}_{x,i+1} = C_{OP}(C_{OO} + \sigma_n^2 I)^{-1}\vec{y}_{i,x} \tag{6}$$

where σ_n^2 is a parameter accounting for noise.

Algorithm 1: Close Sample Prediction

input: (X^O, X^P, K);
H $\leftarrow$ Hyperparameters from CG or RProp
T $\leftarrow$ Average trajectory length in X^O
$Q_1 \leftarrow$ K Nearest Neighbors of $\vec{x}_l^P$ in X^O
$(\Delta\hat{x}_{x,l+1}, \Delta\hat{x}_{y,l+1}, \Delta\hat{x}_{z,l+1}) \leftarrow$ GPR$(\vec{x}_l^P, Q_1|H)$
$\hat{\vec{x}}_{l+1}^P = \vec{x}_l^P + (\Delta\hat{x}_{x,l+1}, \Delta\hat{x}_{y,l+1}, \Delta\hat{x}_{z,l+1})$
for *t = 2:T* **do**
  $Q_t \leftarrow$ K Nearest Neighbors of $\hat{\vec{x}}_{t-1}^P$ in X^O
  $(\Delta\hat{x}_{x,t}, \Delta\hat{x}_{y,t}, \Delta\hat{x}_{z,t}) \leftarrow$ GPR$(\hat{\vec{x}}_{t-1}^P, Q_1|H)$
  $\hat{\vec{x}}_t^P = \hat{\vec{x}}_{t-1}^P + (\Delta\hat{x}_{x,t}, \Delta\hat{x}_{y,t}, \Delta\hat{x}_{z,t})$
end
Return: $(\hat{\vec{x}}_{l+1}^P ... \hat{\vec{x}}_T^P)$

C. Choice of Kernel Function

There are many standard choices of kernel functions for various types of regression problems. For this algorithm, we test with 3 common kernel functions: Squared exponential, Matern 3/2, and Rational Quadratic. We chose these functions as they will always map feature vectors with larger euclidean distances them to lower values. In other words, $||\vec{a} - \vec{b}|| > ||\vec{c} - \vec{d}|| \leftrightarrow f(\vec{a}, \vec{b}) < f(\vec{c}, \vec{d})$.

Specifically, the squared exponential kernel function is:

$$f(\vec{a}, \vec{b}) = \sigma^2 \exp\left(-\frac{||\vec{a} - \vec{b}||^2}{2l^2}\right) \tag{7}$$

where σ and l are hyperparameters. The Matern 3/2 kernel is:

$$f(\vec{a}, \vec{b}) = \sigma^2\left(1 + \sqrt{3}\frac{||\vec{a} - \vec{b}||^2}{2l^2}\right)\exp\left(\sqrt{3} - \frac{||\vec{a} - \vec{b}||^2}{2l^2}\right) \tag{8}$$

Where σ and l are again hyperparameters. Lastly, the rational quadratic kernel function is:

$$f(\vec{a}, \vec{b}) = \sigma^2\left(1 + \frac{||\vec{a} - \vec{b}||^2}{2\alpha l^2}\right)^{-\alpha} \tag{9}$$

where σ, l, and α are all hyperparameters.

D. Hyperparameter Optimization

For any choice of kernel function f, we need to determine a set of hyperparameters to inform the similarity function represented by the kernel. In order to optimize these hyperparameters, we attempt two different methods for gradient-based optimization: Nonlinear Conjugate Gradients (CG) and Resilient Back-propagation (RProp). Each of these methods have been employed for Gaussian Process Regression in the past in [7] and [8] respectively.

Nonlinear Conjugate Gradient methods minimze an arbitrary function $f(x)$ iteratively through:

$$x_{k+1} = x_k + a_k d_k \tag{10}$$

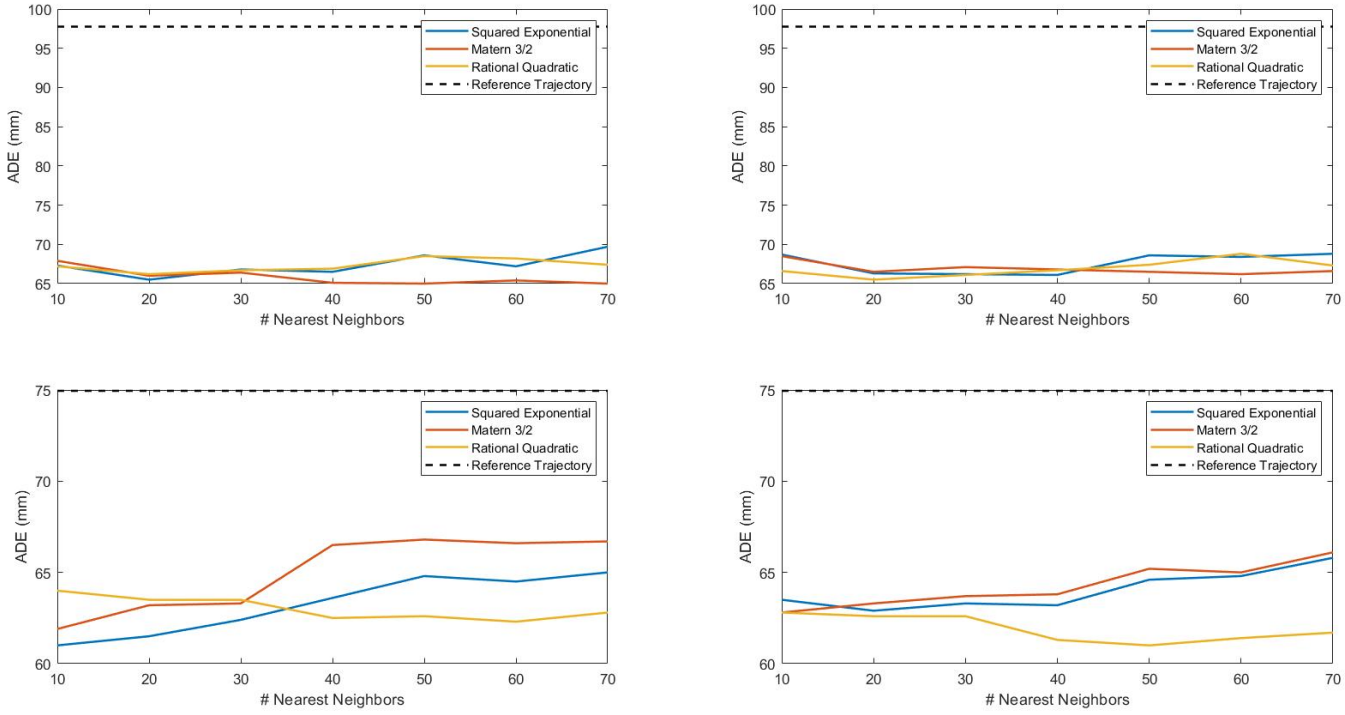

Fig. 1. A comparison of the ADE for different actions and hyperparameters. Each graph shows the ADE vs number of nearest neighbors used in GPR for each kernel function. Each of the four graphs show these data for a combination of action and hyperparameter optimization method. (Top Left) ADE for action 1 using CG optimization. (Top Right) ADE for action 1 using Rprop optimization. (Bottom Left) ADE for action 2 using CG optimization. (Bottom Right) ADE for action 2 using Rprop optimization

Where the stepsize a_k is defined by a line-search and d_k is defined as:

$$d_k = \begin{cases} -\nabla f(x_k) & k = 1 \\ -\nabla f(x_k) + \beta_k d_{k-1} & k \geq 2 \end{cases} \tag{11}$$

We use the Ribiere-Polyak definition of β_k as:

$$\beta_k = \frac{\nabla f(x_k)^T (\nabla f(x_k) - \nabla f(x_{k-1})}{||\nabla f(x_{k-1})||^2} \tag{12}$$

Note that here we estimate gradients via comparison between chosen points in x_k since no symbolic gradient is available.

The RProp algorithm on the other hand only utilizes the sign of a function's gradient, and the update is simply:

$$x_{k+1} = x_k + \eta_k \text{sgn}(\nabla f(x_k)) \tag{13}$$

where

$$\eta_k = \begin{cases} \eta_{k-1} * 1.2 & \nabla f(x_k) * \nabla f(x_{k-1}) > 0 \\ \eta_{k-1} * 0.5 & \nabla f(x_k) * \nabla f(x_{k-1}) < 0 \\ \eta_{k-1} & otheriwse \end{cases} \tag{14}$$

If we wish to optimize the hyperparameters for each temporal iteration of GPR in our prediction, we found the algorithm's run time becomes prohibitively long. However, without much loss of precision we can optimize the hyperparameters for the kernel function over a random subset of points in the trajectories of X^O and use these hyperparameters for each specific instance of GPR applied to the nearest neighbors of a point in question.

E. Complementary Filter with Mean Trajectory

The prediction method described in sections $3A-D$ work to decrease error of predictions in the short term (see section 4 for details on analysis). However, as we attempt to predict further into the future, we see that the errors incurred by our prediction method and the mean trajectory approximately converge. More detrimentally, the IQR of error incurred in our prediction can increase faster with time than the mean trajectory (see Fig 2). We can infer from these observations that as time increases, a good signal for prediction becomes no more reliable than the mean, and the variation subject to a prediction drives the increased IQR. To combat this increase in error IQR, we introduce a complimentary filter with the mean trajectory favoring our prediction method early on and the mean trajectory later. Let the predicted trajectory be $\hat{X}^P$, the mean trajectory be $\bar{X}$, and a function that maps a time step $\bar{X}$ to the DTW aligned time step in X^P be $a(t)$. Our complimentary filter is:

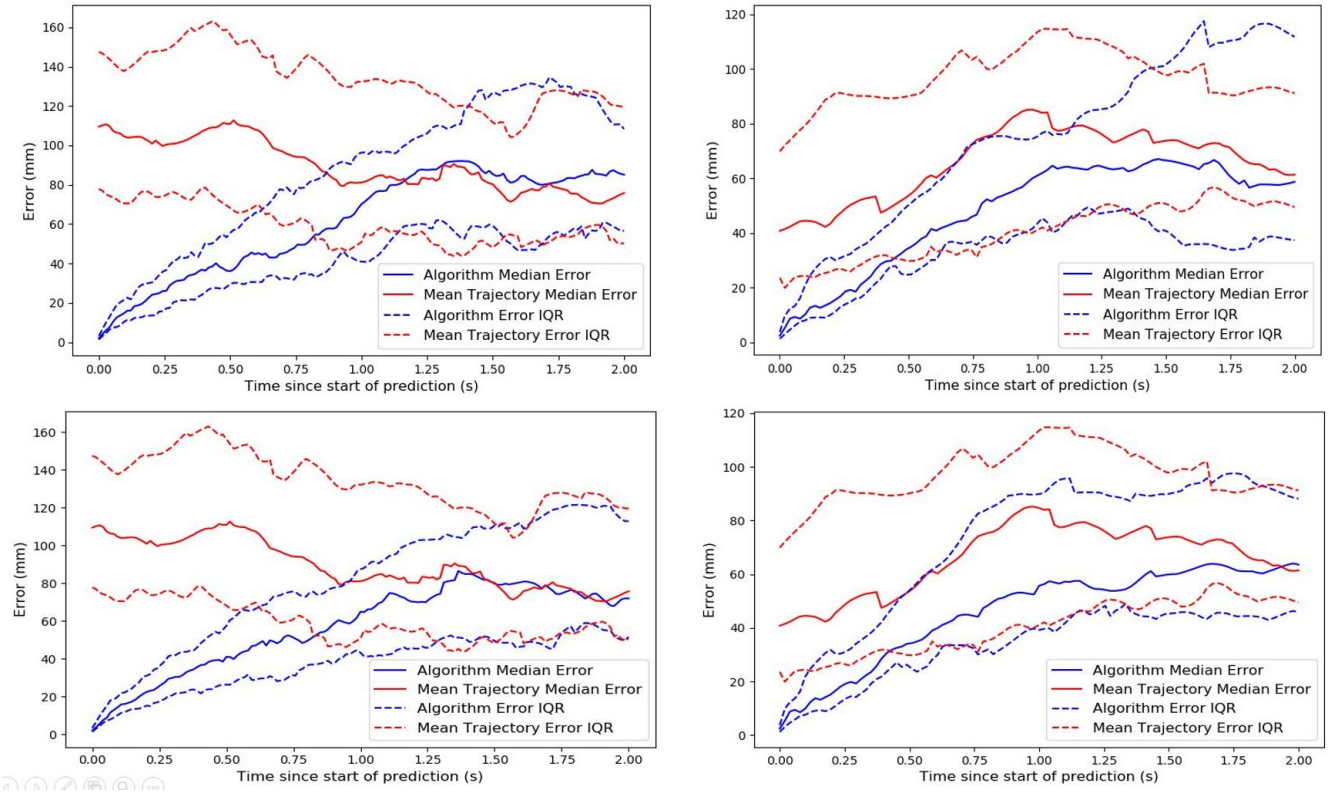

Fig. 2. Euclidian error from ground truth trajectory using our algorithm and the mean trajectory. Each graph shows the errors for a combination of action and usage of complimentary filter. (Top Left) Action 1 without complimentary filter. (Top Right) Action 2 without complimentary filter. (Bottom Left) Action 1 with complimentary filter. (Bottom Right) Action 2 with complimentary filter.

$$\hat{X}^P_{t,filtered} = \sigma(t,\theta)\vec{\bar{x}}_{a(l)+t} + (1-\sigma(t,\theta))\hat{\vec{x}}^P_t \tag{15}$$

where θ are parameters defining the center and width of our sigmoid:

$$\sigma(t,\theta) = \frac{1}{1+e^{-\theta_2(x-\theta_1)}} \tag{16}$$

Currently, we are simply using $\theta_1 = T/2$ and $\theta_2 = 4/T$ where T is the total number of points in our prediction. However, in the future we hope to somehow optimize over these parameters.

IV. Evaluation and Results

To evaluate the algorithm, we compare our performance against a standard baseline in a unique data set. For our analysis, we focus primarily on standard metrics for analyzing human motion prediction across 2 optimization parameters - kernel function and hyperparameter optimization method.

A. Dataset and Methodology

For our analysis, we focus on walked paths during a task where the participant was asked to retrieve several objects from various locations in a room and drop them off one at a time in a predetermined region. We use BEST-PTA to cluster each full trial into segmented paths of particular actions (i.e. exit from one region and entrance into another). To evaluate our algorithm, we use leave-one-out-cross-validation on the action datasets, beginning our prediction 20 samples (.4 seconds) into the trajectory (i.e. X^P has a length of 20).

For this test, a PhaseSpace motion capture system is used to track a participant's head at a rate of 50 Hz. While BEST-PTA segments each trial into many actions, we focus on two actions and one participant. This participant had completed action 1 45 times and action 2 22 times, each taking ≈ 2 seconds to complete.

As the baseline for our analysis, we use the mean trajectory over all observed X^O determined by BEST-PTA. The metric we use to analyze our algorithm is the Average Displacement Error (ADE) defined as the average euclidean distance between the prediction and ground truth trajectory segment over all time steps. Note that we only consider a number of time steps equal to the minimum of the true trajectory and the predicted trajectory. In other words, we only consider the time steps for which there is a one-to-one mapping between the ground truth trajectory and the predicted trajectory.

B. Results

The results of our evaluation give evidence that our algorithm provides an improvement against using the mean trajectory for spatial prediction with respect to ADE. While the two actions have slightly different minimum errors across

optimization method and kernel function, using a squared exponential kernel with 20 nearest neighbors minimizes overall error across both actions. Using these optimal parameters give a 33% lower ADE for action 1 and a 17.9% lower ADE for action 2.

While we can define optimal hyperparameters for the algorithm, the relatively low disparity in ADE across all hyperparameters (as shown in fig 1) is worthy of note. I believe this to be a result of relatively consistent signals in the data resulting from BEST-PTA clustering. Since all trajectories are already segmented and clustered by position, directionality, start, and end points, the data give a very clear signal to the prediction. Therefore, changes in hyperparameters always read near-identical signals from the data and make consistently well-informed predictions.

C. Impact of the Complimentary filter

Investigating Fig 2 provides insight into how our algorithm performs over time with respect to the mean trajectory. In both action 1 and action 2, we see the algorithm give significantly lower error than the mean trajectory for approximately the first 1 to 1.5 seconds of the prediction, at which point the errors approximately converge on the mean trajectory. Additionally, in action 2 we see the IQR of the prediction expand significantly wider than the mean trajectory as the errors of the two converge.

In contrast, when applying the complimentary filter, the prediction for action 2 does not show the same expanding IQR in error. More interestingly, with the complimentary filter applied, we see the ADE decrease in action 1 by 8.0% and by 2.8% in action 2, using the optimal hyperparameters described in section $4B$. This improved error seems to largely result from the complimentary filter allowing our algorithm to increase the time during which we can make predictions better than the mean trajectory. However, more testing and analysis on more actions will be necessary to make any defensible claims on the ADE benefits of a complimentary filter.

REFERENCES

[1] P. Lasota and J. Shah, "Bayesian Estimator for Partial Trajectory Alignment." Robotics: Science and Systems, Breisgau, Germany 2019.

[2] J Elfring, R. Monengraft, M. Steinbuch, "Learning intentions for improved human motion prediction," *Robotics and Autonomous Systems*, vol. 64, pp. 591-602, April 2014

[3] A. Alahi, K. Gel, V. Ramanathan, A. Robicquet, L. Fei-Fei, S Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* pp. 961-971, 2016

[4] M Huynh, G. Alaghband, "Scene-LSTM: A Model for Human Trajectory Prediction", arXiv:1808.04018v2

[5] D. Varshneya, G. Srinivasaraghavan, "Human Trajectory Prediction using Spatially aware Deep Attention Models," arXiv:1705.09436

[6] J. Wiest, M. Höffken, U. Kreßel and K. Dietmayer, "Probabilistic trajectory prediction with Gaussian mixture models," 2012 IEEE Intelligent Vehicles Symposium, Alcala de Henares, pp. 141-146, 2012

[7] A. J. Smola, P. L. Bartlett, "Sparse greedy Gaussian process regression." *In Advances in neural information processing systems*, pp. 619-625, 2001

[8] M Blum, R. A. Martin, Blum, M. and Martin A. Riedmiller. "Optimization of Gaussian process hyperparameters using Rprop." ESANN (2013)." *The European Symposium on Artificial Neural Networks* 2013