Robot Learning

Gaussian Process Regression

Prof. Songhwai Oh
ECE, SNU

[Reference]

Gaussian Processes for Machine Learning, C. E. Rasmussen and C. K. I. Williams, MIT Press, 2006.

GAUSSIAN PROCESS REGRESSION (GPR)

Gaussian Process Regression

- $\mathcal{X}$: index set (e.g., time $\mathbb{R}$, space $\mathbb{R}^3$)
- $z(x)$: a collection of random variables with $x \in \mathcal{X}$.
- $z(x)$ is a **Gaussian process** if for any finite set $\{x_1, \ldots, x_n\}$, $\{z(x_1), \ldots, z(x_n)\}$ has a multivariate Gaussian distribution, with mean $\mu \in \mathbb{R}^n$ and covariance $K \in \mathbb{R}^{n \times n}$.
- The mean μ and covariance K depend on the chosen finite set $\{x_1, \ldots, x_n\}$.
- **Gaussian process regression**: A nonparametric regression method using properties of Gaussian processes.
- Two views to interpret Gaussian process regression:
  - Weight-space view
  - Function-space view

WEIGHT-SPACE VIEW

Linear Regression

- $f(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$
- $y(\mathbf{x}) = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.
- Suppose we have collected n input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$.
- Define $X = [\mathbf{x}_1^T; \ldots; \mathbf{x}_n^T]^T$. Then

$$P(\mathbf{y}|X, \mathbf{w}) = \prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w}) \quad (1)$$

$$= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\left(-\frac{1}{2\sigma_n^2}\|\mathbf{y} - X^T\mathbf{w}\|^2\right) \quad (2)$$

$$= \mathcal{N}(X^T\mathbf{w}, \sigma_n^2\mathbb{I}). \quad (3)$$

- **Linear regression**: Find $\mathbf{w}$ such that $\|\mathbf{y} - X^T\mathbf{w}\|^2$ is minimized.
  - Solution: $\widehat{\mathbf{w}} = \left(XX^T\right)^{-1} X\mathbf{y}$

Weight-Space View: Bayesian Formulation

- $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$
- $y(\mathbf{x}) = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.
- Suppose we have collected n input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$.
- Define $X = [\mathbf{x}_1^T; \ldots; \mathbf{x}_n^T]^T$. Then

$$
\begin{aligned}
P(\mathbf{y}|X, \mathbf{w}) &= \prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w}) && (1)\\
&= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\left(-\frac{1}{2\sigma_n^2}\|\mathbf{y} - X^T\mathbf{w}\|^2\right) && (2)\\
&= \mathcal{N}(X^T\mathbf{w}, \sigma_n^2 \mathbb{I}). && (3)
\end{aligned}
$$

- **Bayesian formulation**: Put a prior over the parameters, i.e., $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$.

$$
p(\mathbf{w}|\mathbf{y}, X) = \frac{P(\mathbf{y}|X, \mathbf{w})P(\mathbf{w})}{P(\mathbf{y}|X)}
$$

Weight-Space View: Posterior Distribution

$$
\begin{aligned}
P(\mathbf{w}|\mathbf{y}, X) &= \frac{P(\mathbf{y}|X, \mathbf{w})P(\mathbf{w})}{P(\mathbf{y}|X)} \\
&\propto \exp\left(-\frac{1}{2\sigma_n^2}(\mathbf{y} - X^T\mathbf{w})^T(\mathbf{y} - X^T\mathbf{w})\right)\exp\left(-\frac{1}{2}\mathbf{w}^T\Sigma_p^{-1}\mathbf{w}\right) \\
&\propto \exp\left(-\frac{1}{2}(\mathbf{w} - \bar{\mathbf{w}})^T\left(\frac{1}{\sigma_n^2}XX^T + \Sigma_p^{-1}\right)(\mathbf{w} - \bar{\mathbf{w}})\right),
\end{aligned}
$$

where $\bar{\mathbf{w}} = \frac{1}{\sigma_n^2}\left(\frac{1}{\sigma_n^2}XX^T + \Sigma_p^{-1}\right)^{-1}X\mathbf{y}$. Hence,

$$
\begin{aligned}
P(\mathbf{w}|\mathbf{y}, X) &= \mathcal{N}(\bar{\mathbf{w}}, A^{-1}) \\
\bar{\mathbf{w}} &= \frac{1}{\sigma_n^2}\left(\frac{1}{\sigma_n^2}XX^T + \Sigma_p^{-1}\right)^{-1}X\mathbf{y} = \frac{1}{\sigma_n^2}A^{-1}X\mathbf{y} \\
A &= \left(\frac{1}{\sigma_n^2}XX^T + \Sigma_p^{-1}\right).
\end{aligned}
$$

Note that the MAP estimate $\widehat{\mathbf{w}}_{\mathrm{MAP}}$ for $\mathbf{w}$ is $\bar{\mathbf{w}}$.

Weight-Space View: Predictive Distribution

Suppose that we want to predict at a new input $\mathbf{x}_*$, then the predictive distribution of $f_* = f_*(\mathbf{x}_*)$ is (by integrating out $\mathbf{w}$):

$$\begin{aligned} P(f_*|\mathbf{x}_*, X, \mathbf{y}) &= \int P(f_*|\mathbf{x}_*, \mathbf{w}) P(\mathbf{w}|X, \mathbf{y}) d\mathbf{w} \\ &= \mathcal{N}\left(\frac{1}{\sigma_n^2}\mathbf{x}_*^T A^{-1} X\mathbf{y}, \mathbf{x}_*^T A^{-1}\mathbf{x}_*\right), \end{aligned}$$

where $A = \left(\frac{1}{\sigma_n^2} XX^T + \Sigma_p^{-1}\right)$.

Weight-Space View: Kernel Trick (1)

- $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^N$, a mapping from the input space to the high-dimensional feature space ($N \gg D$).
- $f(\mathbf{x}) = \phi(\mathbf{x})^T w$.
- Define $\Phi(X) = [\phi(\mathbf{x}_1)^T; \ldots; \phi(\mathbf{x}_n)^T]^T \in \mathbb{R}^{N \times n}$. Then

$$f_* | \mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}\left(\frac{1}{\sigma_n^2}\phi(\mathbf{x}_*)^T A^{-1}\Phi\mathbf{y}, \phi(\mathbf{x}_*)^T A^{-1}\phi(\mathbf{x}_*)\right),$$

  where $A = \frac{1}{\sigma_n^2}\Phi\Phi^T + \Sigma_p^{-1}$ and $A \in \mathbb{R}^{N \times N}$.
- Before the transformation the dimension of A was $D \times D$. But now, it is $N \times N$, which is much bigger. There is an issue with computing the inverse of this new A.
- Let $K = \Phi^T \Sigma_p \Phi \in \mathbb{R}^{n \times n}$ and $\phi_* = \phi(\mathbf{x}_*)$.

Weight-Space View: Kernel Trick (2)

Let $K = \Phi^T \Sigma_p \Phi$ and $\phi_* = \phi(\mathbf{x}_*)$ and consider (recall $A = \sigma_n^{-2} \Phi\Phi^T + \Sigma_p^{-1}$)

$$\begin{aligned} A\Sigma_p\Phi &= \left(\sigma_n^{-2}\Phi\Phi^T + \Sigma_p^{-1}\right)\Sigma_p\Phi = \sigma_n^{-2}\Phi\Phi^T\Sigma_p\Phi + \Phi \\ &= \sigma_n^{-2}\Phi\left(\Phi^T\Sigma_p\Phi + \sigma_n^2\mathbb{I}\right) = \sigma_n^{-2}\Phi\left(K + \sigma_n^2\mathbb{I}\right). \end{aligned}$$

Premultiply A^{-1} and postmultiply $\left(K + \sigma_n^2\mathbb{I}\right)^{-1}$ to get

$$\begin{aligned} \sigma_n^{-2}A^{-1}\Phi\left(K + \sigma_n^2\mathbb{I}\right)\left(K + \sigma_n^2\mathbb{I}\right)^{-1} &= A^{-1}A\Sigma_p\Phi\left(K + \sigma_n^2\mathbb{I}\right)^{-1} \\ \sigma_n^{-2}A^{-1}\Phi &= \Sigma_p\Phi\left(K + \sigma_n^2\mathbb{I}\right)^{-1} \end{aligned}$$

Hence, the predictive mean becomes

$$\sigma_n^{-2}\phi_*^T A^{-1}\Phi\mathbf{y} = \phi_*^T\Sigma_p\Phi\left(K + \sigma_n^2\mathbb{I}\right)^{-1}\mathbf{y}.$$

Using the matrix inversion lemma, we can also show that the predictive covariance matrix becomes

$$\phi_*^T A^{-1}\phi_* = \phi_*^T\Sigma_p\phi_* - \phi_*^T\Sigma_p\Phi\left(K + \sigma_n^2\mathbb{I}\right)^{-1}\Phi^T\Sigma_p\phi_*.$$

Weight-Space View: Kernel Trick (3)

Substitute new terms to get

$$f_*|\mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}(\phi_*^T \Sigma_p \Phi \left(K + \sigma_n^2 \mathbb{I}\right)^{-1} \mathbf{y},$$
$$\phi_*^T \Sigma_p \phi_* - \phi_*^T \Sigma_p \Phi \left(K + \sigma_n^2 \mathbb{I}\right)^{-1} \Phi^T \Sigma_p \phi_*).$$

- Note that all terms are inner products between ϕ's and the inversion is over a matrix of size $n \times n$, instead of $N \times N$. In general, we can expect $n < N$.
- We can now apply the kernel trick and replace $\phi(\mathbf{x})^T \Sigma_p \phi(\mathbf{x}')$ by $k(\mathbf{x}, \mathbf{x}')$.

FUNCTION-SPACE VIEW

Function-Space View

- $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$, i.e., $f(x)$ is a Gaussian process
- $m(x) = \mathbb{E}(f(x))$, mean function
- $k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]$, covariance function
- Example: $f(x) = \phi(x)^T w$ with $w \sim \mathcal{N}(0, \Sigma_p)$.
  - $\mathbb{E}(f(x)) = \phi(x)^T \mathbb{E}(w) = 0$.
  - $\mathbb{E}(f(x)f(x')) = \phi(x)^T \mathbb{E}(ww^T)\phi(x') = \phi(x)^T \Sigma_p \phi(x')$.
  - Hence, $f(x)$ and $f(x')$ are jointly Gaussian.
  - It is also true for $f(x_1), \ldots, f(x_n)$ for any $x_1, \ldots, x_n$ and n.
  - Therefore, $f(x)$ is a Gaussian process.
- If $K(x_p, x_q) = \mathbf{cov}(f(x_p), f(x_q))$, then (assuming $m(x) = 0$)

$$f_* \sim \mathcal{N}(0, K(x_*, x_*)).$$

Function-Space View: Prediction w/o Noise

f and f_* are jointly Gaussian, hence, for any finite number of measurements at $x_1, \ldots, x_n$ and x_*, (again assuming $m(x) = 0$)

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{pmatrix}\right),$$

where $[K(X,X)]_{ij} = k(x_i, x_j)$.

Recall that the conditional distribution of a jointly Gaussian random vector $[\mathbf{x}^T \ \mathbf{y}^T]^T$ is such that $\mathbf{x}|\mathbf{y} \sim \mathcal{N}(\mathbb{E}(\mathbf{x}|\mathbf{y}), \Sigma_{x|y})$, where

$$\mathbb{E}(\mathbf{x}|\mathbf{y}) = \mathbb{E}(\mathbf{x}) + \Sigma_{xy}\Sigma_{yy}^{-1}(\mathbf{y} - \mathbb{E}(\mathbf{y})) \quad (1)$$

$$\Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}. \quad (2)$$

By conditioning, we get

$$f_*|X_*, X, f \sim \mathcal{N}(K(X_*,X)K(X,X)^{-1}f, \\ K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*))$$

Function-Space View: Prediction w/ Noise

Let $y(x) = f(x) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.

Then $\mathbf{cov}(y(x_p), y(x_q)) = K(x_p, x_q) + \sigma_n^2 \delta_{pq}$ or in a matrix form

$$\mathbf{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 \mathbb{I}$$

The joint distribution between y and f_* is

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(X,X) + \sigma_n^2 \mathbb{I} & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{pmatrix}\right)$$

By conditioning, we get

$$f_* | X, \mathbf{y}, X_* \sim \mathcal{N}(\bar{f}_*, \mathrm{cov}(f_*)),$$

$$\bar{f}_* = K(X_*, X)\left(K(X,X) + \sigma_n^2 \mathbb{I}\right)^{-1} \mathbf{y}$$

$$\mathbf{cov}(f_*) = K(X_*, X_*) - K(X_*, X)\left(K(X,X) + \sigma_n^2 \mathbb{I}\right)^{-1} K(X, X_*)$$

Function-Space View: Linear Predictor

Let $k_* = K(X, X_*)$ and $K = K(X, X)$. Then

$$\bar{f}_* = k_*^T \left(K + \sigma_n^2 \mathbb{I}\right)^{-1} \mathbf{y}$$
$$\text{cov}(f_*) = K(X_*, X_*) - k_*^T \left(K + \sigma_n^2 \mathbb{I}\right)^{-1} k_*$$

We have a "linear predictor" (linear combination of $\mathbf{y}$) since

$$\bar{f}_* = \sum_{i=1}^{n} \alpha_i k(x_i, x_*),$$

where $\alpha = \left(K + \sigma_n^2 \mathbb{I}\right)^{-1} \mathbf{y}$.

More on GPR

Learning

Since $\mathbf{y} \sim \mathcal{N}(0, K + \sigma_n^2 \mathbb{I})$, the log marginal likelihood is

$$\log P(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2\mathbb{I})^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2\mathbb{I}| - \frac{n}{2}\log 2\pi,$$

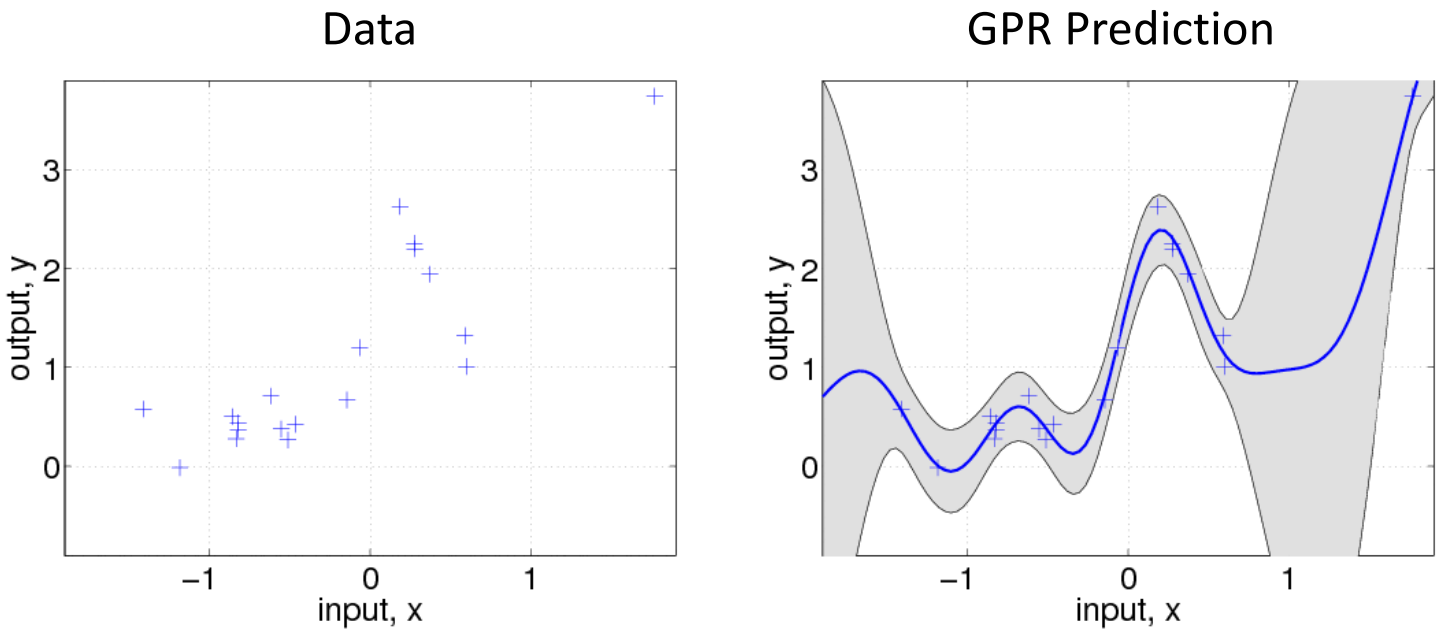
which can be used to estimate σ_n^2 and parameters for the kernel function (using a gradient based method).

For example, if the following squared exponential kernel is used, the kernel parameters are (σ_f^2, σ_l^2).

$$K(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2\sigma_l^2}\|x_p - x_q\|^2\right)$$

In practice, selecting the right kernel for a given problem is also an important task.

Comments on GPR



- **Pros:** principled, probabilistic, predictive uncertainty
- **Cons:** computationally intensive ($n \times n$ matrix inversion)
  - GPR uses all data: $f(x_*) = \sum_{i=1}^{n} \alpha_i k(x_i, x_*)$, where $\alpha = \left(K + \sigma_n^2 \mathbb{I}\right)^{-1} \mathbf{y}$
  - cf. SVM is sparse: $f(x_*) = \sum_{i \in \mathcal{S}} a_i k(x_i, x_*) + b$

Kriging

- **Kriging** is a well-known method in geostatistics (mining), meteorology, and statistics since 1960's (by G. Matheron inspired from the Master's thesis by D.G. Krige).
- Best linear unbiased prediction.

Suppose we are given data $\mathbf{y} = [y_1 \ \ldots \ y_n]^T$ taken from locations $x_1, \ldots, x_n$, and want to predict its value at a new location x_*. (Assume the mean is zero.)

- Linear predictor: $\mathbf{w}^T\mathbf{y}$ for some weight vector $\mathbf{w}$
- Best linear unbiased predictor minimizes the mean squared prediction error: $\mathbb{E}\left(f(x_*) - \mathbf{w}^T\mathbf{y}\right)^2$.
- The optimal predictor is

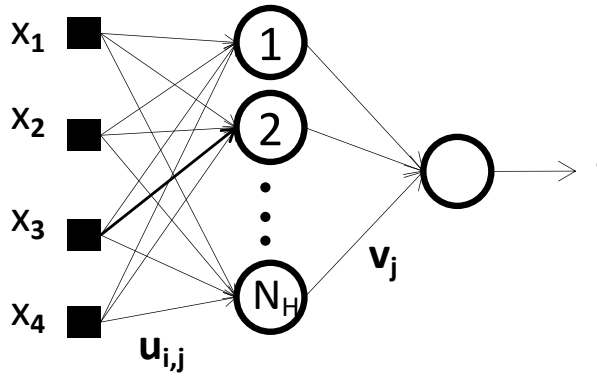
$$\bar{f}_* = \mathbf{cov}(f_*, \mathbf{y})\mathbf{cov}(\mathbf{y})^{-1}\mathbf{y} = K(X_*, X)\left(K(X, X) + \sigma_n^2\mathbb{I}\right)^{-1}\mathbf{y}.$$

- The corresponding mean squared prediction error is

$$\begin{aligned}&\mathbf{var}(f_*, f_*) - \mathbf{cov}(f_*, \mathbf{y})\mathbf{cov}(\mathbf{y})^{-1}\mathbf{cov}(\mathbf{y}, f_*)\\ &\qquad = K(X_*, X_*) - K(X_*, X)\left(K(X, X) + \sigma_n^2\mathbb{I}\right)^{-1}K(X, X_*)).\end{aligned}$$

Neural Networks

- Neural network with a single hidden layer with N_H units



$$f(\mathbf{x}) = b + \sum_{j=1}^{N_H} v_j h(\mathbf{x}; \mathbf{u}_j)$$

[Cybenko 1989, Hornik 1993]

- Neural network with one hidden layer is a **universal approximator** as $N_H \to \infty$
- That is, it can approximate any continuous function on a compact support under mild conditions.

Cybenko, G. (1989). **Approximation by superpositions of a sigmoidal function**. Mathematics of Control, Signals and Systems, 2(4):303-314.
Hornik, K. (1993). **Some New Results on Neural Network Approximation**. Neural Networks, 6(8):1069–1072.

Neural Network Converges to a GP

Suppose that

- $b \sim (0, \sigma_b^2)$ and $v_j \sim (0, \sigma_v^2)$
- $\mathbf{u}_j$ are independently and identically distributed
- σ_v^2 scales as ω^2 / N_H

$$f(\mathbf{x}) = b + \sum_{j=1}^{N_H} v_j h(\mathbf{x}; \mathbf{u}_j)$$

$$\begin{aligned} \mathbb{E}(f(\mathbf{x})) &= 0 \\ \mathbb{E}\left(f(\mathbf{x}) f(\mathbf{x}')\right) &= \sigma_b^2 + \sum \sigma_v^2 \mathbb{E}_{\mathbf{u}}\left(h(\mathbf{x}; \mathbf{u}_j) h(\mathbf{x}'; \mathbf{u}_j)\right) \\ &= \sigma_b^2 + \omega^2 \mathbb{E}_{\mathbf{u}}\left(h(\mathbf{x}; \mathbf{u}_j) h(\mathbf{x}'; \mathbf{u}_j)\right) \end{aligned}$$

[Neal 1996] By the central limit theorem, $f(\mathbf{x})$ converges to a Gaussian process as $N_H \to \infty$.

If $h(\mathbf{x}; \mathbf{u}) = \text{erf}(u_0 + \sum u_j x_j)$ and $\mathbf{u} \sim \mathcal{N}(0, \Sigma)$, then the covariance function of the neural network is

$$k_{NN}(\mathbf{x}, \mathbf{x}') = \frac{2}{\pi} \sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^T \Sigma \tilde{\mathbf{x}}'}{((1 + 2\tilde{\mathbf{x}}^T \Sigma \tilde{\mathbf{x}})(1 + 2\tilde{\mathbf{x}}'^T \Sigma \tilde{\mathbf{x}}'))^{1/2}}\right),$$

where $\tilde{\mathbf{x}} = [1 \; x_1 \; \ldots \; x_d]^T$.

Neal, R. M. (1996). **Bayesian Learning for Neural Networks**. Springer, New York. Lecture Notes in Statistics 118.

Summary

Gaussian process regression:

- Weight-space view: Bayesian approach to linear regression (with the kernel trick)
- Function-space view: MMSE estimate, linear predictor
- Best linear unbiased prediction (kriging)

- Provides the predictive variance for an unseen data
- Can be computationally intensive (for prediction, $O(n^3)$)

- A single hidden layer neural network converges to a Gaussian process

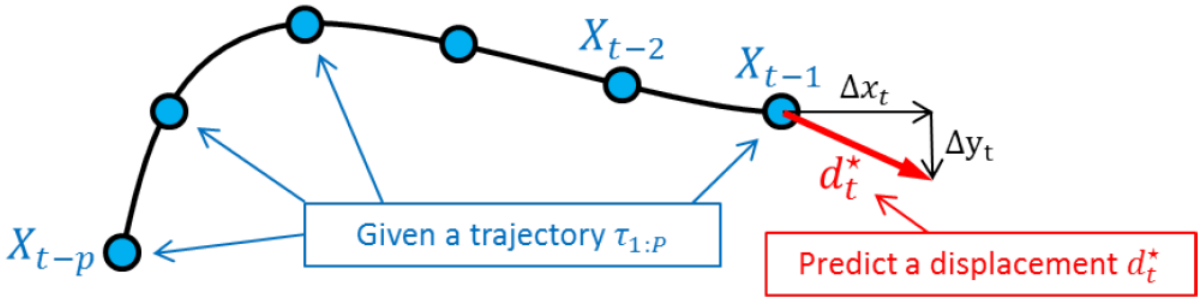
Real-Time Autonomous Robot Navigation

Sungjoon Choi, Eunwoo Kim, Kyungjae Lee, Songhwai Oh, "**Real-Time Nonparametric Reactive Navigation of Mobile Robots in Dynamic Environments**," Robotics and Autonomous Systems, vol. 91, pp. 11–24, May 2017.
Sungjoon Choi, Eunwoo Kim, Kyungjae Lee, and Songhwai Oh, "**Leveraged Non-Stationary Gaussian Process Regression for Autonomous Robot Navigation**," in Proc. of the IEEE International Conference on Robotics and Automation (ICRA), May 2015.

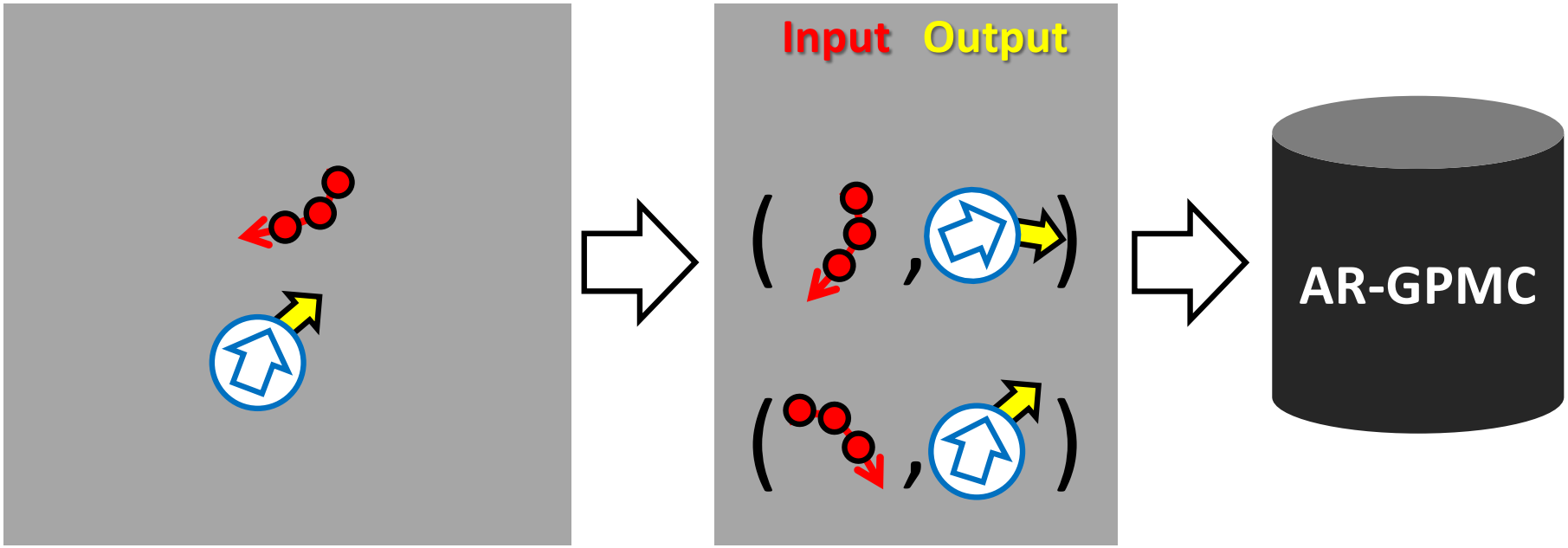
Knightscope K5 Meets a Guy

Real-Time Navigation

Autoregressive Gaussian Process Motion Model



Autoregressive Gaussian Process Motion Controller



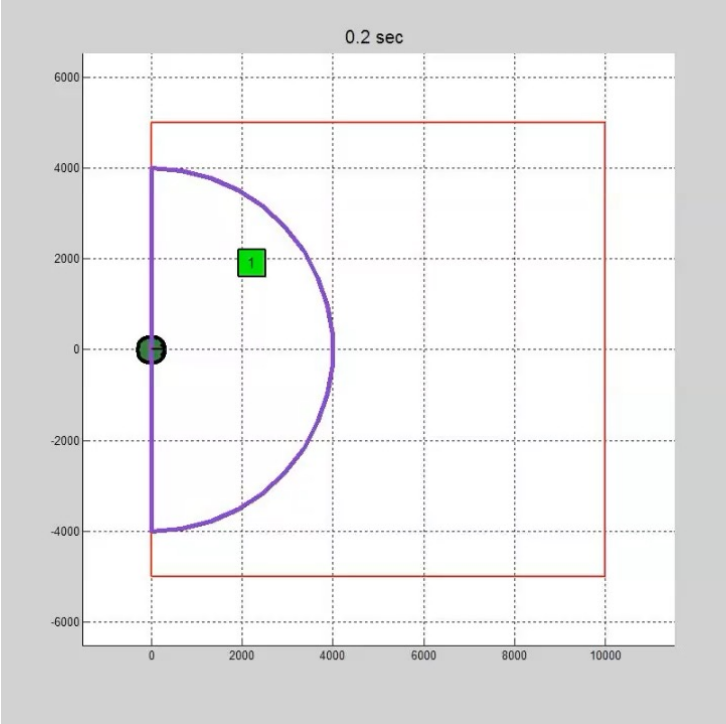
Control a robot to avoid an incoming object (with sample based MPC)

Collect state-action pairs from diverse scenarios

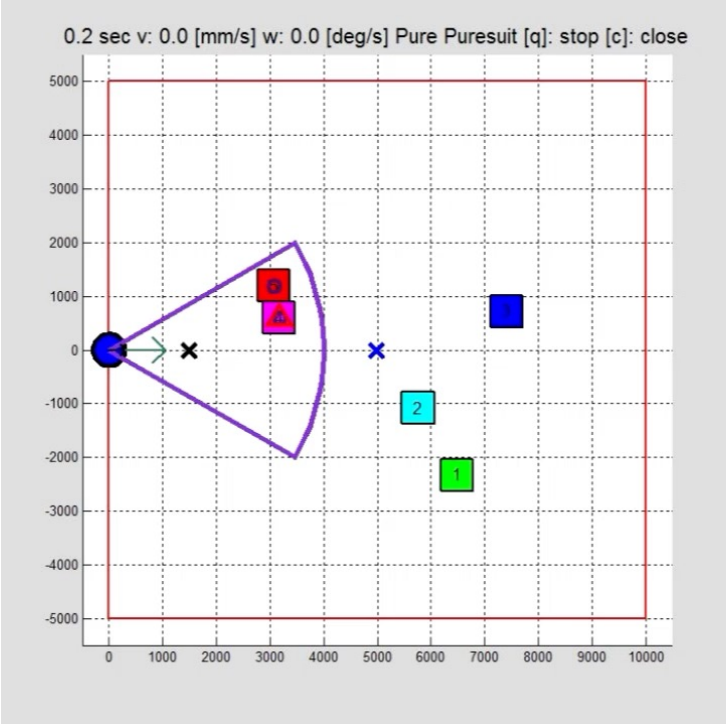
These data will be served as a motion controller

Real-Time Navigation

Training Phase (slow)



Execution Phase (fast)



(Trajectory, Control) pairs are collected from diverse scenarios. This is a time-consuming work.

Learned motion controller is used in the execution phase.
It takes less than 50ms to make one control.

Real-Time Navigation

Selecting the Number of Training Data

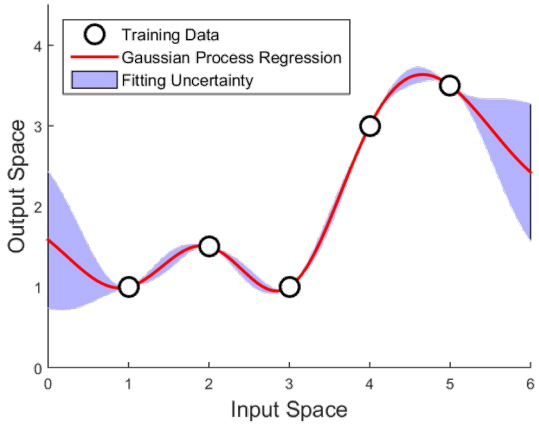
Under the assumption that a target function f is drawn from a Gaussian process with known covariance function $k(\mathbf{x}, \mathbf{x}')$ and we observe n d-dimensional locations of f, resulting a training set $\mathcal{D}_n = (\mathbf{X}, \mathbf{y})$ of size n, where $\mathbf{X}$ are locations and $\mathbf{y}$ are outputs of the training set, the generalization error is given by

$$E^g(f) = \int E^g_{\mathbf{X}}(f) d\mathbf{X}, \tag{13}$$

where

$$E^g_{\mathbf{X}}(f) = \int \left(k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^T \mathbf{K}(\mathbf{X})^{-1} \mathbf{k}(\mathbf{x}_*)\right) p(\mathbf{x}_*) d\mathbf{x}_*, \tag{14}$$

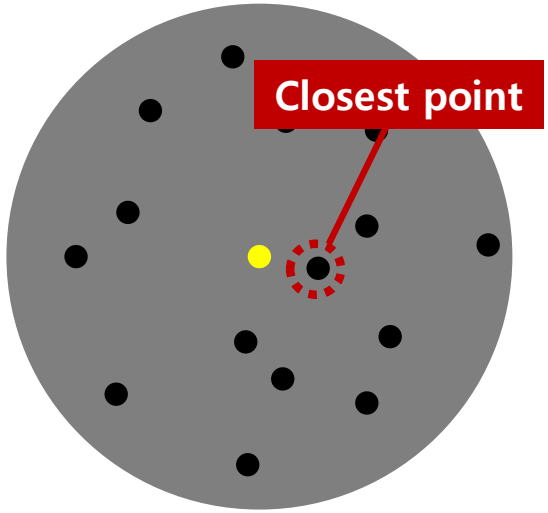
Gaussian process predictive variance



Lemma 1. *The generalization error (13) for Gaussian process regression satisfies*

$$E^g(f) \leq E^g_U(f) = \int \int \sigma^2_{\mathbf{x}_c}(\mathbf{x}_*) p(\mathbf{X}) p(\mathbf{x}_*) d\mathbf{X} d\mathbf{x}_*, \tag{18}$$

where $\mathbf{x}_c$ is a training input which is nearest to $\mathbf{x}_*$, $\mathbf{X}$ is a set of n training inputs, $p(\mathbf{x}_*)$ and $p(\mathbf{X})$ are probability density functions of a test input and a set of training inputs, respectively, and $\sigma^2_{\mathbf{x}_c}(\mathbf{x}_*) = \sigma^2_f(1 - \bar{k}(\mathbf{x}_*, \mathbf{x}_c)^2)$ is the predictive variance computed using only one training data $\mathbf{x}_c$.

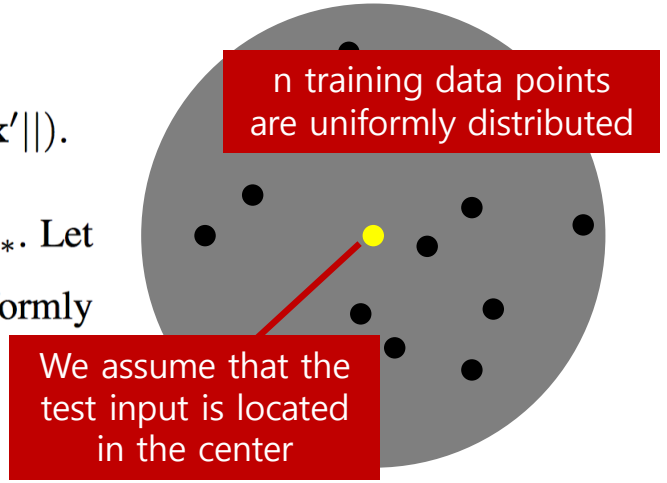


Real-Time Navigation

Selecting the Number of Training Data

With two following assumptions:

- Assumption 1: The kernel function is isotropic, i.e., $k(\mathbf{x}, \mathbf{x}') = k_I(||\mathbf{x} - \mathbf{x}'||)$.
- Assumption 2: The training data points are uniformly distributed around $\mathbf{x}_*$. Let R be the radius of a ball around $\mathbf{x}_*$, such that there are n data points uniformly placed inside the ball.

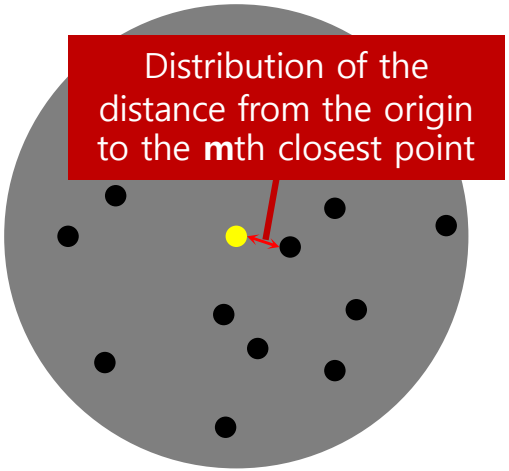


And combined with following theorem:

Theorem 1 ([29]). *In a binomial point process (BPP) consisting of n points uniformly and randomly distributed in a d-dimensional ball of radius R centered at the origin, the Euclidean distance R_m from the origin to its mth nearest point follows a generalized beta distribution, i.e., for $r \in [0, R]$,*

$$f_{R_m}(r) = \frac{d}{R}\frac{B(m - 1/d + 1, n - m + 1)}{B(n - m + 1, m)} \times \beta\left(\left(\frac{r}{R}\right)^d; m - \frac{1}{d} + 1, n - m + 1\right),$$

where $\beta(x; a, b) = (1/B(a, b))x^{a-1}(1 - x)^{b-1}$ and B is the beta function.



Real-Time Navigation

Selecting the Number of Training Data

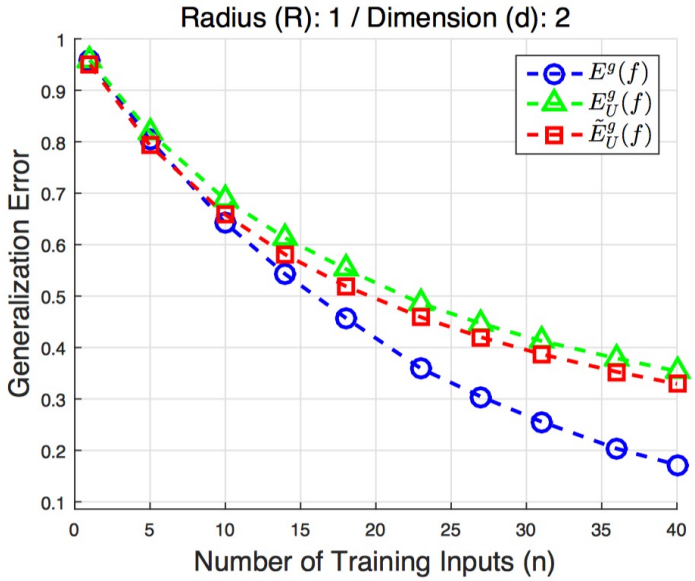
We can efficiently approximate the upper bound of the learning curve of Gaussian process regression as follows:

As an isotropic kernel function is assumed, predictive variance is a function of a distance.

$$E_U^g(f) \approx \tilde{E}_U^g(f) = \int_0^R \sigma_I^2(r) f_{R_1}(r) dr$$

Distance distribution from a binomial point process.

Two integrals with respect to N training data configurations X and test input location $x_\star$ become one integral with respect to r.



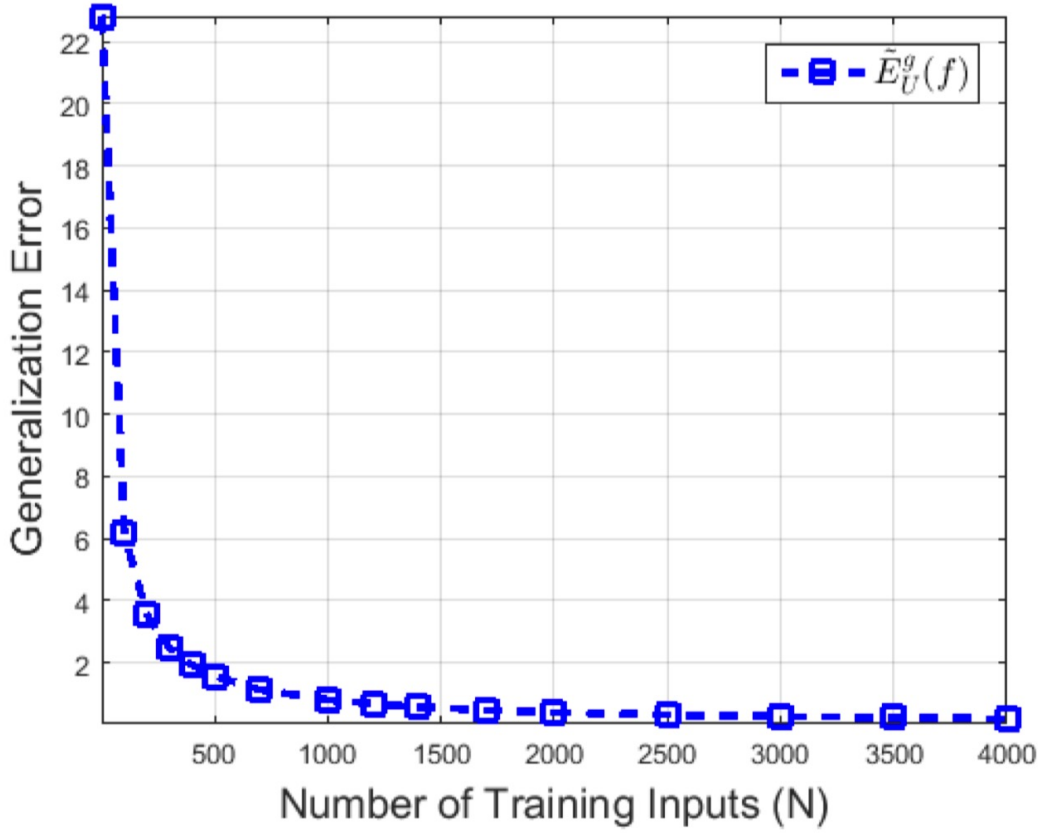
Green: Original upper-bound

Red: Proposed upper-bound

Blue: Learning curve

Real-Time Navigation

Selecting the Number of Training Data

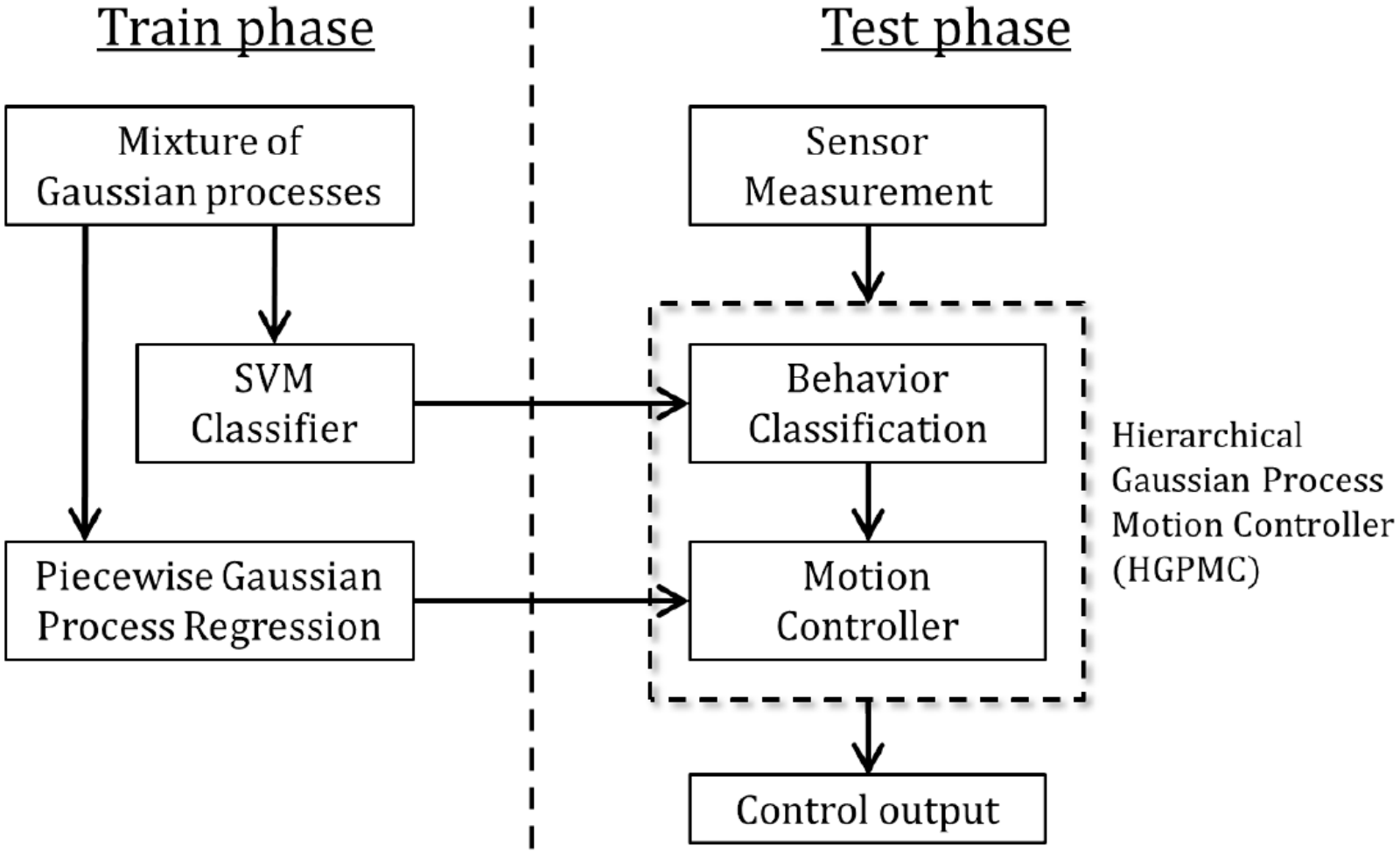


Radius is 5 / dimension is 6

The volume of integration is reduced from $[0, R]^{6^{n+1}}$ to $[0, R]$.

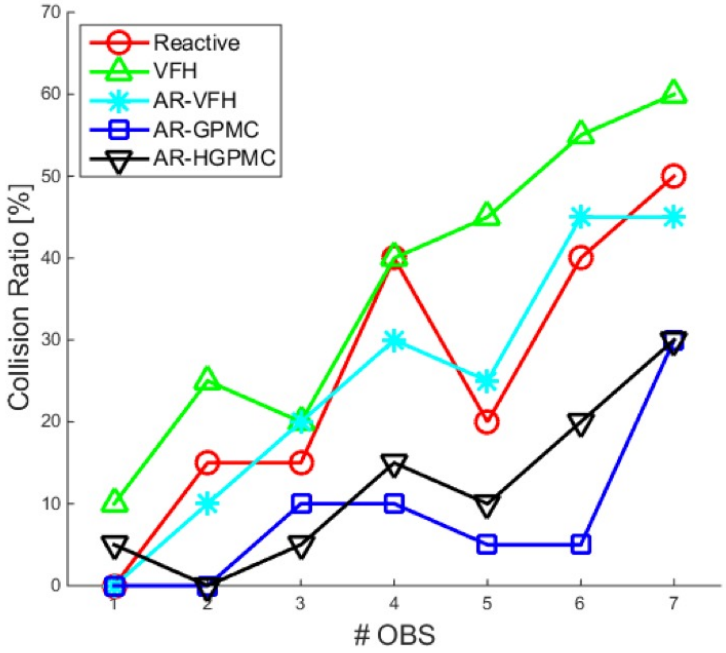
Real-Time Navigation

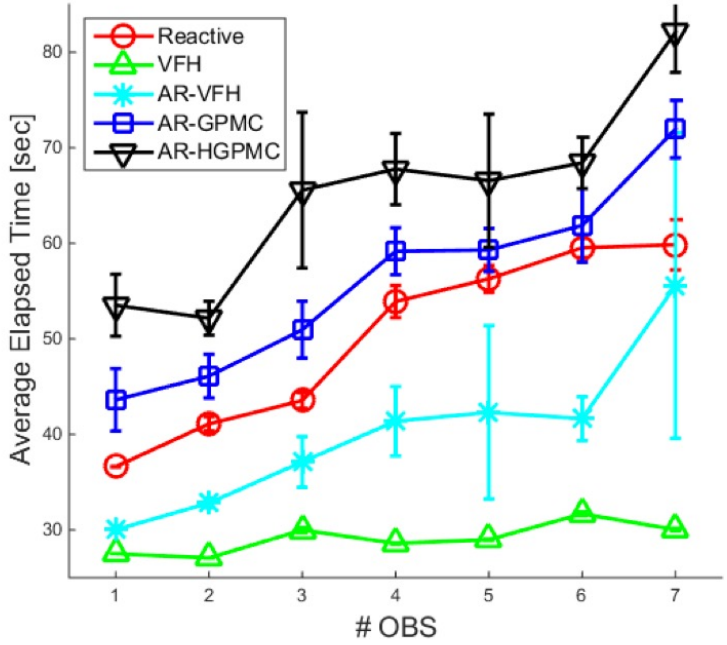
Hierarchical Motion Controller (Mixture of Experts)



Real-Time Navigation

Experiments





	Reactive	VFH	AR-VFH	GPMC	HGPMC
Collision Rate (%)	25.7	36.4	25	**8.57**	12.14
Computation Time (ms)	112.26	**0.91**	1.25	11.9	3.23
Collision Velocity (cm/s)	10	19.6	17.9	12.4	**0.47**

QuadCore 2.7GHz