Gaussian Process Regression

Yubo Chen
2014011058
Dept. of E.E. Tsinghua University

Abstract—Regression analysis is one of the basic research problems in statistics, signal processing, machine learning and other fields. Gaussian processes regression (GPR) is a popular regression analysis method in recent years, which is based on modeling with Gaussian processes (GP). Previous works have shown that GPR yields better performance than other regression analysis methods in many situations. However, GPR's performance depends on the choice of kernel function, likelihood function and inference method. Fortunately, there has been automated model construction method for GPR, named Structure Search (SS). In this article, I compare influences of different kernel functions-including kernels generated by SS-on one-dimensional regression. Similarly, I compare influences of different likelihood functions and inference methods on multi-dimensional regression. It is worth mentioning that I did a small trick by adding a white noise kernel when building kernel functions, which can significantly improve regression performance. According to works mentioned above, I finally choose SS-generated kernel to perform one-dimensional regression and I choose Warped Gaussian likelihood function and Laplace approximation with squared exponential kernel (with Automatic Relevance Determination (ARD) distance measure) to solve the plane control problem. Experiment results show that these models can yield significant performance on provided data.

Index Terms—Gaussian Process Regression, Structure Search, Kernel Functions, likelihood Functions, Inference Methods.

1 Introduction

Regression analysis is one of the basic research problems in statistics, signal processing, machine learning and other fields, which studies relationship between variables, for example, $y = f(\mathbf{x}) + e$, in which $\mathbf{x}$ and y denotes variables and e denotes error random variables. Gaussian processes regression (GPR) is a popular regression analysis method in recent years. The basic idea of GPR is to model $\{f(\mathbf{x})|\mathbf{x} \in \mathrm{S}\}$ with Gaussian process (GP) $\mathcal{N}\{m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')\}$, in which $\mathbb{E}\{f(\mathbf{x})\} = m(\mathbf{x})$ and $\mathrm{cov}(f(\mathbf{x}), f(\mathbf{x}')) = K(\mathbf{x}, \mathbf{x}')$. Given training dataset with noise and the probability density function (pdf) of e, one can estimate the posterior distribution of $f(\mathbf{x}^*)$ at any other point $\mathbf{x}^*$. Previous works show that Gaussian processes regression outperformed in some problems compared with other regression methods such as support vectors [8] [14].

It is not necessary to specify the form of the regression function directly in the GPR method, but it is necessary to determine the function form of the kernel function $K(\mathbf{x}, \mathbf{x}')$. Substantially the kernel function $K(\mathbf{x}, \mathbf{x}')$ gives the correlation between two random variables $f(\mathbf{x})$ and $f(\mathbf{x}')$, so the specified $m(\mathbf{x})$ and $K(\mathbf{x}, \mathbf{x}')$ determines the generalization ability of Gaussian process model. Previous works show that there have been impressive automated method for model construction called Structure Search, which use a greedy algorithm to find kernel function automatically [2] [3] [12].

In addition to kernel functions, likelihood functions and inference methods have some influence on the performance of Gaussian process regression. Generally the posterior distribution is implemented by Bayesian inference. With Gaussian likelihood function, the posterior distribution of $f(\mathbf{x}^*)$ is also Gaussian distribution, which is analytically tractable. However, the implementation of Bayesian inference calls for the evaluation of several integrals, which may not be analytically tractable. In general one may have resort to analytical approximations, for example Laplace approximation and Variational Bayesian approximation [18], or Markov Chain Monte Carlo (MCMC) methods.

The model described above is general GP model. In recent years some researchers presented some new models, such as Additive Gaussian Process [4], Warped Gaussian Process [5], Bayesian Warped Gaussian Process [13] et.al. These models have more complicated forms, which can model more complicated functional correlations. Previous works show that these model can perform much better than general GP model in particular situations [4] [7] [10].

In this article I compared influences of several different kernels on regression. After that I constructed the GP model with Structure Search method. What's more, I did a small trick by adding a white noise kernel to the SS-generated kernel, which can significantly improve the regression performance. Also, I did simulations through MATLAB to compare different models, different likelihood functions and inference methods. Finally I use the best simulation result to solve the one-dimensional

problem and the multi-dimensional plane control problem.

This article is structured as follows: Section 2 is a detailed description of Gaussian Processes Regression; Section 3 describes three model construction methods implemented in our work. Section 4 will introduce the two problems in this project and the methods I used to solve these two problems. Section 5 will show the experimental results and Section 6 is the conclusion.

2 Gaussian Processes Regression

GAUSSIAN processes (GP) is a special kind of stochastic process, which is a generalization of Gaussian probability distribution. To be precise, a GP is any distribution over functions such that any finite set of function values $f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)$ have a joint Gaussian distribution. A GP model, before conditioning on data, is completely specified by its *mean function*,

$$\mathbb{E}\{f(\mathbf{x})\} = m(\mathbf{x}) \tag{2.1}$$

and its *kernel function*, also known as *covariance function*:

$$\text{cov}\{f(\mathbf{x}), f(\mathbf{x}')\} = K(\mathbf{x}, \mathbf{x}') \tag{2.2}$$

then we will write the Gaussian process as:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')) \tag{2.3}$$

The Gaussian processes regression (GPR) is a regression method using GP models [14]. Before introducing GPR, we first introduce the standard linear regression model. Then the introduction of GPR will be based on it.

2.1 The Standard Linear Model

The standard linear regression model with Gaussian noise is defined as follows:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} \qquad y = f(\mathbf{x}) + e \tag{2.4}$$

where $\mathbf{x}$ denotes the input vector, $\mathbf{w}$ is a vector of weights of the linear model, f is the function value and y is the observed target value. We assume that the observed values y differs from the function values $f(\mathbf{x})$ by additive Gaussian noise:

$$e \sim \mathcal{N}(0, \sigma_n^2) \tag{2.5}$$

This noise assumption with the model directly gives rise to the likelihood, which is factored over cases in the training set to give

$$\begin{aligned} p(\mathbf{y}|X, \mathbf{w}) &= \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \mathbf{w}) \\ &= \prod_{i=1}^{n} \frac{1}{(2\pi\sigma_n^2)^{1/2}} \exp(-\frac{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}{2\sigma_n^2}) \\ &= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp(-\frac{1}{2\sigma_n^2}|\mathbf{y} - X^\top \mathbf{w}|^2) \\ &\sim \mathcal{N}(X^\top \mathbf{w}, \sigma_n^2 I) \end{aligned} \tag{2.6}$$

In the Bayesian formalism we need to specify a prior over the parameters, expressing our beliefs about the prior parameters before we look at the observations. We put a zero mean Gaussian prior with covariance matrix Σ_p on the weights: $\mathbf{w} \sim \mathcal{N}(0, \Sigma_p)$. Inference in the Bayesian linear model is based on the posterior distribution posterior over the weights, computed by Bayes' rule:

$$\begin{aligned} \text{posterior} &= \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \\ p(\mathbf{w}|\mathbf{y}, X) &= \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)} \end{aligned} \tag{2.7}$$

where the normalizing constant, also known as the marginal likelihood is given by

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})d\mathbf{w} \tag{2.8}$$

Writing only the terms from the likelihood and prior which depend on the weights, we obtain

$$\begin{aligned} p(\mathbf{w}|\mathbf{y}, X) &\propto \exp(-\frac{1}{2\sigma_n^2}|\mathbf{y} - X^\top \mathbf{w}|^2)\exp(-\frac{1}{2}\mathbf{w}^\top \Sigma_p^{-1}\mathbf{w}) \\ &\propto \exp(-\frac{1}{2}(\mathbf{w} - \bar{\mathbf{w}})^\top (\frac{1}{\sigma_n^2} XX^\top + \Sigma_p^{-1})(\mathbf{w} - \bar{\mathbf{w}})) \end{aligned} \tag{2.9}$$

where $\bar{\mathbf{w}} = \sigma_n^{-2}(\sigma_n^{-2}XX^\top + \Sigma_p^{-1})^{-1}X\mathbf{y}$. Easily we recognize the Gaussian form of posterior distribution:

$$p(\mathbf{w}|\mathbf{y}, X) \sim \mathcal{N}(\bar{\mathbf{w}} = \frac{1}{\sigma_n^2} A^{-1} X\mathbf{y}, A^{-1}) \tag{2.10}$$

where $A = \sigma_n^{-2}XX^\top + \Sigma_p^{-1}$. To make predictions for a test case one can average over all possible parameter values, weighted by their posterior probability. The predictive distribution for $f_* \triangleq f(\mathbf{x}_*)$ at $\mathbf{x}_*$ is given by averaging the output of all possible linear models with respect to the Gaussian posterior

$$\begin{aligned} p(f_*|\mathbf{x}_*, X, \mathbf{y}) &= \int p(f_*|\mathbf{x}_*, \mathbf{w})p(\mathbf{w}|X, \mathbf{y})d\mathbf{w} \\ &= \mathcal{N}(\frac{1}{\sigma_n^2}\mathbf{x}_*^\top A^{-1} X\mathbf{y}, \mathbf{x}_*^\top A^{-1}\mathbf{x}_*) \end{aligned} \tag{2.11}$$

2.2 Generalization of Linear Model

We introduce the function $\phi(\mathbf{x})$ which maps a D-dimensional input vector $\mathbf{x}$ into an N dimensional feature space. Let the matrix $\Phi(X)$ be the aggregation of columns $\phi(\mathbf{x})$ for all cases in the training set. Now the model is

$$f(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w} \tag{2.12}$$

According to (2.11) the predictive distribution becomes

$$f_*|\mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}(\frac{1}{\sigma_n^2}\phi(\mathbf{x}_*)^\top A^{-1}\Phi\mathbf{y}, \phi(\mathbf{x}_*)^\top A^{-1}\phi(\mathbf{x}_*)) \tag{2.13}$$

where $\Phi = \Phi(X)$ and $A = \sigma_n^{-2}\Phi\Phi^\top + \Sigma_p^{-1}$. To make predictions using this equation we need to invert the A matrix of size $N \times N$ which may not be convenient if N, the dimension of the feature space, is large. However, we can rewrite the equation in the following way

$$\begin{aligned} f_*|\mathbf{x}_*, X, \mathbf{y} \sim \mathcal{N}(&\phi_*^\top \Sigma_p \Phi(K + \sigma_n^2 I)^{-1}\mathbf{y}, \\ &\phi_*^\top \Sigma_p \phi_* - \phi_*^\top \Sigma\Phi(K + \sigma_n^2 I)^{-1}\Phi^\top \Sigma_p \phi_*) \end{aligned} \tag{2.14}$$

where $\phi(\mathbf{x}_*) = \phi_*$ and $K = \Phi^\top \Sigma_p \Phi$. It can be easily proved that (2.13) and (2.14) is equivalent actually [14]. Notice that the feature space always enters in the form of $\Phi^\top \Sigma_p \Phi, \phi_*^\top \Sigma_p \Phi$, or $\phi_*^\top \Sigma_p \phi_*$, we can define kernel function $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}')$. As Σ_p is positive definite we can define $\Sigma_p^{1/2}$ so that $(\Sigma_p^{1/2})^2 = \Sigma_p$, then defining $\psi(\mathbf{x}) = \Sigma_p^{1/2}\phi\mathbf{x}$ we can get $K(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x}) \cdot \psi(\mathbf{x}')$, which have the form of covariance function. Notice that $f(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w}$ with prior $\mathbf{w} \sim \mathcal{N}(0, \Sigma_p)$, we have the mean and covariance as follows:

$$\begin{aligned} \mathbb{E}\{f(\mathbf{x})\} &= \phi(\mathbf{x})^\top \mathbb{E}\{\mathbf{w}\} = 0 \\ \mathbb{E}\{f(\mathbf{x})f(\mathbf{x}')\} &= \phi(\mathbf{x})^\top \mathbb{E}\{\mathbf{w}\mathbf{w}^\top\}\phi(\mathbf{x}') = \phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}) \end{aligned} \tag{2.15}$$

Thus $f(\mathbf{x})$ and $f(\mathbf{x}')$ are jointly Gaussian with zero mean and covariance given by $\phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x})$. Indeed, the function values $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)$ corresponding to any number of input points n are jointly Gaussian, which reminds us of Gaussian process models.

2.3 Regression with GP Models

From previous section we know that a GP model with a specific kernel can describe relationships between input vectors and output values, which means that through kernel functions we can calculate the predictive distribution $f_*|\mathbf{x}_*, X, \mathbf{y}$. Using predictive distribution one can yield prediction on any input vectors. Generally there are two cases of prediction: prediction with noise-free observations and prediction with noisy observations. Here we only focus on the latter one.

First we define matrix $K(X, X)$: $K(X, X)_{ij} \triangleq K(x_i, x_j) = \mathrm{cov}(x_i, x_j)$ where $X \triangleq (x_1, x_2, \ldots, x_n)$. Then we assume that the kernel function is already selected and the observations have additive independent identically distributed Gaussian noise ε. The relationship between input and output can be modeled as $y = f(\mathbf{x}) + \varepsilon$. The prior on the noisy observations becomes

$$\begin{aligned} \mathrm{cov}(y_p, y_q) &= K(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \\ \mathrm{cov}(\mathbf{y}) &= K(X, X) + \sigma_n^2 I \end{aligned} \tag{2.16}$$

Introducing the noise term we can write the joint distribution of the observed target values and the function values (values without noise) at the test locations under the prior as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right). \tag{2.17}$$

Deriving the conditional distribution we arrive at the *key predictive equations* for Gaussian processes regression

$$f_*|X_*, X, \mathbf{y} \sim \mathcal{N}(\bar{f}_*, \mathrm{cov}(f_*)) \tag{2.18}$$

$$\bar{f}_* = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y} \tag{2.19}$$

$$\begin{aligned} \mathrm{cov}(f_*) = &K(X_*, X_*) \\ &- K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X_*). \end{aligned} \tag{2.20}$$

For any set of basis functions, we can compute the corresponding covariance function as $K(\mathbf{x}_p, \mathbf{x}_q) = \phi(\mathbf{x}_p)^\top \Sigma_p \phi(\mathbf{x}_q)$; conversely for every positive definite covariance function K, there exists a expansion in terms of basis functions. Another way to look at this equation is to see 2.19 as a linear combination of n kernel functions, each one centered on a training point, by writing

$$\bar{f}(\mathbf{x}_*) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}_*) \tag{2.21}$$

where $\mathbf{x}_*$ is test point and $\alpha = (K(X, X) + \sigma_n^2 I)^{-1}\mathbf{y}$.

Although the GP defines a joint Gaussian distribution over all of the y variables, one for each point in the index set $\mathcal{X}$, for making predictions at $\mathbf{x}_*$ we only care about the $(n+1)$-dimensional distribution defined by the n training points and the test point. As a Gaussian distribution is marginalized by just taking the relevant block of the joint covariance matrix it is clear that conditioning this $(n+1)$-dimensional distribution on the observations gives the desired result. Then we should introduce the *marginal likelihood* $p(\mathbf{y}|X)$ at this point. The marginal likelihood is the integral of the likelihood times the prior

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f}. \tag{2.22}$$

Under the Gaussian process model the prior is Gaussian, $\mathbf{f}|X \sim \mathcal{N}(0, K)$ and the likelihood is a factorized Gaussian $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 I)$ so we can perform the integration yielding the log marginal likelihood. A practical implementation of Gaussian processes regression(GPR)is shown in Algorithm 1.

Algorithm 1 Prediction for GP Regression

procedure GPRegression(X, y, $K(\cdot,\cdot)$, σ_n^2, $\mathbf{x}_*$)
  $L := \mathrm{cholesky}(K + \sigma_n^2 I)$
  $\alpha := L^\top \backslash (L \backslash y)$
  $\bar{f}_* := \mathrm{k}_*^\top \alpha$ ▷ predictive mean
  $\mathrm{v} := L \backslash \mathrm{k}_*$
  $\mathbb{V}[f_*] := K(\mathbf{x}_*, \mathbf{x}_*) - \mathrm{v}^\top \mathrm{v}$ ▷ predictive variance
  $\log p(\mathbf{y}|X) := \log \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f}$
  return: $\bar{f}_*,\ \mathbb{V}[f_*],\ \log p(\mathbf{y}|X)$
end procedure

3 Model Construction

As described in previous section, we can perform regression with a given kernel function. However, the selection of kernel function is of great importance on the performance of regression, which represents the correlation between input data. In this section we introduce three methods of constructing GP model implemented in this project: Section 3.2 gives a detailed introduction of Structure Search; Section 3.3 is a description of Additive Gaussian Process method; Section 3.4 gives a introduction of Warped Gaussian Process. Before introducing these three methods, we introduce the basis theory on model selection in Section 3.1.

3.1 Model Selection

In previous section we introduced how to do regression using a Gaussian processes with given kernel. However, in many practical applications, it may not be easy to determine kernel functions with confidence. In order for a model to be a practical tool in an application, one needs to make decisions about the details of its specification. A multitude of possible families of covariance functions exists, each of these families typically have a number of free *hyperparameters* whose values also need to be determined. Choosing a covariance function for a particular application thus comprises both comparing across different families and setting of hyperparameters within a family.

Generally we use Bayesian model selection methods. It is common to use a hierarchical specification of models. At the lowest level are the parameters (weights), $\mathbf{w}$. At the second level are hyperparameters $\boldsymbol{\theta}$ which control the distribution of the parameters at the bottom level. At the top level are model structures $\mathcal{H}_i$. At the bottom level, the *posterior* over the parameters is given by Bayes' rule

$$p(\mathbf{w}|\mathbf{y}, X, \boldsymbol{\theta}, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)}{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)} \tag{3.1}$$

where $p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)$ is the *likelihood* and $p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)$ is the parameter *prior*. The normalizing constant in the denominator of (3.1) $p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)$ is independent of the parameters, called *marginal likelihood*, and is given by

$$p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i) = \int p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)d\mathbf{w}. \tag{3.2}$$

At the second level, the Bayes' rule is similar:

$$p(\boldsymbol{\theta}|\mathbf{y}, X, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)}{p(\mathbf{y}|X, \mathcal{H}_i)}, \tag{3.3}$$

and the marginal likelihood is given by

$$p(\mathbf{y}|X, \mathcal{H}_i) = \int p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)d\boldsymbol{\theta}. \tag{3.4}$$

At the top level the posterior for the model have the same form:

$$p(\mathcal{H}_i|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y}|X)}, \tag{3.5}$$

where $p(\mathbf{y}|X) = \sum_i p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i)$. Notice that the implementation of Bayesian inference calls for the evaluation of several integrals, which may or may not be analytically tractable. So one may have to resort to analytical approximations or Markov Chain Monte Carlo (MCMC) methods. Several common approximation methods, also called inference methods, are Laplace approximation, Variational Bayesian approximation, Expectation Propagation approximation et. al. The marginal likelihood is valuable in solving the model selection problem because it has the property of automatically incorporating a trade-off between model fit and model complexity. In order to set the hyperparameters, we have to maximizing the marginal likelihood.

3.2 Structure Search

Before maximizing the marginal likelihood to set hyperparameters, the chosen of mean function and kernel function is also important, which corresponds to $\mathcal{H}_i$. The selection of mean function is relatively easy, as marginalizing over an unknown mean function can be equivalently expressed as a zero-mean GP with a new kernel [3]. So the core problems converts to the selection of kernel function [19]. It has already been proved that positive semidefinite kernels are closed under addition and multiplication [9]. This allows one to create richly structured and interpretable kernels from well understood base component. By summing kernels, we can model data as a superposition of independent functions, possibly representing different structures. By multiplying kernels one can account for interactions between different input dimensions or different notions of similarity.

Structure Search is an impressive way of automatically create kernel function. In this method we use four well-understood base kernel families: squared exponential (SE), periodic (PER), linear (LIN), and rational quadratic (RQ). Any algebraic expression combining these kernels using the operations + and × defines a kernel family, whose parameters are the concatenation of the parameters for the base kernel families. The search procedure begins by proposing all base kernel families applied to all input dimensions. The following search operators over the set of expressions are allowed:

(1) Any subexpression $\mathcal{S}$ can be replaced with $\mathcal{S}+\mathcal{B}$, where $\mathcal{B}$ is any base kernel family.
(2) Any subexpression $\mathcal{S}$ can be replaced with $\mathcal{S} \times \mathcal{B}$, where $\mathcal{B}$ is any base kernel family.
(3) Any base kernel $\mathcal{B}$ may be replaced with any other base kernel family $\mathcal{B}'$.

These operators can generate all possible algebraic expressions. Structure Search algorithm searches over algebraic expression kernel space using a greedy search: at each stage, choose the highest scoring kernel and expand it by applying all possible operators. Figure 1 shows an example of search tree followed by this algorithm. In particular,

- One can look for structure in the residuals of a model, and then extend the model to capture that structure. This corresponds to applying rule (1).
- One can start with structure which is assumed to hold globally, but find that it only holds locally. This corresponds to applying rule (2).
- One can add features incrementally, analogous to algorithms like boosting, backfitting, or forward selection. This corresponds to applying rules (1) or (2) to dimensions not yet included in the model.

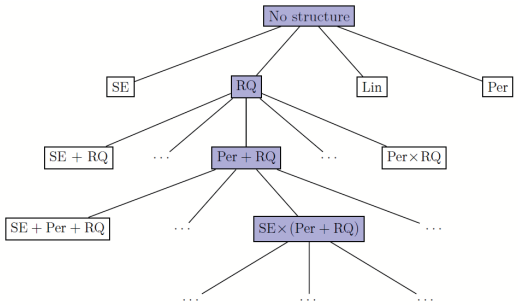

Fig. 1: An example of a search tree over kernel expressions.

Choosing kernel structure requires a criterion for evaluating structures. One can choose marginal likelihood as criterion, since it balances the fit and complexity of a model.

3.3 Additive Gaussian Process

Before the advent of the Structure Search algorithm, there was also a regression method (kernel function) designed specifically for high dimensional data. We first assign each dimension $i \in \{1 \dots D\}$ a one-dimensional *base kernel* $K_i(x_i, x_i')$. We then define the first order, second order and nth order additive kernel as:

$$K_{add_1}(\mathbf{x}, \mathbf{x}') = \sigma_1^2 \sum_{i=1}^{D} K_i(x_i, x_i') \tag{3.6}$$

$$K_{add_2}(\mathbf{x}, \mathbf{x}') = \sigma_2^2 \sum_{i=1}^{D} \sum_{j=i+1}^{D} K_i(x_i, x_i') K_j(x_j, x_j') \tag{3.7}$$

$$K_{add_n}(\mathbf{x}, \mathbf{x}') = \sigma_n^2 \sum_{1 \le i_1 \le \cdots \le i_n \le D} \left[\prod_{d=1}^{n} K_{i_d}(x_{i_d}, x_{i_d}') \right] \tag{3.6}$$

where D is the dimension of input space and σ_n^2 is the variance assigned to all nth order interactions. The nth covariance function is a sum of $\binom{D}{n}$ terms. In particular, the Dth order additive covariance function has $\binom{D}{D} = 1$ term, a product of each dimension¡¯s covariance function:

$$K_{add_D}(\mathbf{x}, \mathbf{x}') = \sigma_D^2 \prod_{d=1}^{D} K_d(x_d, x_d') \tag{3.7}$$

The full additive kernel is a sum of the additive kernels of all orders.

An An additive kernel over D inputs with interactions up to order n has $O(2^n)$ terms,which makes naively summing over these terms intractable. Fortunately, the nth order additive kernel corresponds to the nth *elementary symmetric polynomial*, and the Newton-Girard formulae give an efficient recursive form for computing these polynomials [11]. Conveniently, we can use this trick to efficiently compute all of the necessary derivatives of the additive kernel with respect to the base kernels. This trick allows us to optimize the base kernel hyperparameters with respect to the marginal likelihood. Also, as experiment results show, typically only the first few orders of interaction are important for modeling a given function, which means that one can simply limit the maximum degree of interaction without losing much accuracy.

3.4 Warped Gaussian Process

In this section we consider a popular choice of kernel functions, in which case the entries in the covariance matrix are given by

$$C_{mn} = v_1 \exp\left[-\frac{1}{2} \sum_{d=1}^{D} \left(\frac{x_d^{(m)} - x_d^{(n)}}{r_d} \right)^2 \right] + v_0 \delta_{mn} \tag{3.8}$$

where r_d is a width parameter expressing the scale over which typical functions vary in the dth dimension, v_1 is a size parameter expressing the typical size of the overall process in y-space, v_0 is the noise variance of the observations, and $\mathbf{\Theta} = \{v_0, v_1, r_1, \dots, r_D\}$. It is simple to show that the predictive distribution for a new point given the observed data, $P(t_{N+1}|\mathbf{t}_N, \mathbf{X}_{N+1})$, is Gaussian. Learning, or ¡®training¡¯, in a GP is usually achieved by finding a local maximum in the likelihood using conjugate gradient methods with respect to the hyperparameters $\mathbf{\Theta}$ of the covariance matrix. The negative log likelihood is given by

$$\begin{aligned} \mathcal{L} &= -\log P(\mathbf{t}_N|\mathbf{X}_N, \mathbf{\Theta}) \\ &= \frac{1}{2} \log \det \mathbf{C}_N + \frac{1}{2} \mathbf{t}_N^\top \mathbf{C}_N^{-1} \mathbf{t}_N + \frac{N}{2} \log 2\pi. \end{aligned} \tag{3.9}$$

Let us consider a vector of latent targets $\mathbf{z}_N$ and suppose that this vector is modelled by a GP,

$$\begin{aligned} \mathcal{L} &= -\log P(\mathbf{z}_N|\mathbf{X}_N, \mathbf{\Theta}) \\ &= \frac{1}{2} \log \det \mathbf{C}_N + \frac{1}{2} \mathbf{z}_N^\top \mathbf{C}_N^{-1} \mathbf{z}_N + \frac{N}{2} \log 2\pi. \end{aligned} \tag{3.10}$$

Now we make a transformation from the true observation space to the latent space by mapping each observation through the same monotonic function f,

$$z_n = f(t_n; \mathbf{\Psi}) \qquad \forall n, \tag{3.11}$$

where $\mathbf{\Psi}$ parameterises the transformation. We require f to be monotonic and mapping on to the whole of the real line; otherwise probability measure will not be conserved in the transformation, and we will not induce a valid distribution over the targets $\mathbf{t}_N$. Including the Jacobian term that takes the transformation into account, the negative log likelihood, $-\log P(\mathbf{t}_N|\mathbf{X}_N, \mathbf{\Theta}, \mathbf{\Psi})$, now becomes:

$$\begin{aligned} \mathcal{L} = &\frac{1}{2} \log \det \mathbf{C}_N + \frac{1}{2} f(\mathbf{t}_N)^\top \mathbf{C}_N^{-1} \mathbf{t}_N \\ &- \sum_{n=1}^{N} \log \left. \frac{\partial f(t)}{\partial t} \right|_{t_n} + \frac{N}{2} \log 2\pi \end{aligned} \tag{3.12}$$

Learning in this extended model is achieved by simply taking derivatives of the negative log likelihood function (3.12) with respect to both $\mathbf{\Theta}$ and $\mathbf{\Psi}$ parameter vectors, and using a conjugate gradient method to compute ML parameter values.

For a particular setting of the covariance function hyperparameters $\mathbf{\Theta}$, in *latent* variable space the predictive distribution at a new point is just as for a regular GP:

$$P(z_{N+1}|\mathbf{x}^{(N+1)}, \mathcal{D}, \mathbf{\Theta}) = \mathcal{N}(\hat{z}_{N+1}(\mathbf{\Theta}), \sigma^2_{N+1}(\mathbf{\Theta})). \quad (3.13)$$

To find the distribution in the observation space we pass that Gaussian through the nonlinear warping function, giving

$$P(t_{N+1}|\mathbf{x}^{(N+1)}, \mathcal{D}, \mathbf{\Theta}, \mathbf{\Psi}) = \frac{f'(t_{N+1})}{\sqrt{2\pi\sigma^2_{N+1}}} \exp\left[-\frac{1}{2}\left(\frac{f(t_{N+1}) - \hat{z}_{N+1}}{\sigma_{N+1}}\right)^2\right]. \quad (3.14)$$

The shape of this distribution depends on the form of the warping function f, but in general it may be asymmetric and multimodal.

We wish to design a warping function that will allow for complex transformations, but we must constrain the function to be monotonic. An obvious way to do this is a neural-net style sum of tanh functions,

$$f(t_n; \mathbf{\Psi}) = t + \sum_{i=1}^{I} a_i \tanh(b_i(t + c_i)) \quad a_i, b_i \geq 0 \quad \forall i \quad (3.15)$$

where $\mathbf{\Psi} = \{\mathbf{a,b,c}\}$[13]. This produces a series of smooth steps, with **a** controlling the size of the steps, **b** controlling their steepness, and **c** their position. Of course the number of steps I needs to be set, and that will depend on how complex a function one wants.

4 Problems and Methods

THE methods mentioned above are all theoretical methods. In practical applications, we need to compare the various methods to choose the best one. In this section the methods of solving realistic problems in this project are introduced. In this project we have two problems:

- There are 543 train data and 87 test data in question1.mat. Using train data to train GPR machine to predict values on test data. Compare the prediction values and real values and calculate mean-square error.
- There are some data from the plane control problem of F16 in question2.mat. In xtrain and xtest every data have 40 dimensions, which describes 40 types of plane state. Using GPR to solve the plane control problem and perform predictions on xtest. Compare the prediction values and real values and calculate mean-square error.

The following sections respectively describes the methods of solving the two problems.

4.1 One-dimensional Regression

For one-dimensional regression problem, the choice of kernel function is the core problem. First, I carried out MATLAB simulation on several other relatively simple kernel functions, including LIN+PER and Polynomial+PER+Constant. Then in order to achieve the best results, I implemented the Structure Search algorithm to select the kernel function, which gives a more complicated kernel function: LIN×SE+SE×(PER+RQ).

It is worth mentioning that considering noise in train data, I add a White Noise (WN) kernel to the Struture-Search-generated kernel function. Surprisingly, this small trick yield significant better performance. Detailed numerical experiment results will be described in Section 5.1.

4.2 Multi-dimensional Regression

For multi-dimensional regression, in addition to the choice of kernel function, the choice of likelihood function and inference method also affect the performance of regression model. First, I tried SE-based additive kernel. In order to test the performance of different orders of additive kernel, I set the maximum order to 40 and decrease one order per step. Until the order goes down to only one, this kernel gives its best performance. Considering that with order equals to one the additive kernel is equivalent to sum of base kernels, I tried other SE kernel functions, and finally find the best one: SE with Automatic Relevance Determination.

After the kernel function is determined, the next step is to choose likelihood function and inference method. Since the likelihood function describes the probability distribution of the error between the observed value ($\mathbf{y} = f(\mathbf{x})+e$) and the function value ($f(\mathbf{x})$) which is based on data, there is no good method so far to select the likelihood function except the simulation experiment. Inference method is used to approximate integral results in calculating marginal likelihood using Bayesian inference, so the choice of inference methods can directly affect the accuracy of regression results. In order to compare the differences between combinations of likelihood functions and inference methods, I selected several of them and simulated all the combinations. Note that Warped Gaussian Process also act as a likelihood function, which outperforms than other likelihood when combining with Laplace approximation. Detailed numerical experiment results will be described in Section 5.2.

Kernel	MSE
LIN+PER	46.5174
POLY+PER+CONST	46.5090
LIN×SE+SE×(PER+RQ)	0.17335
LIN×SE+SE×(PER+RQ)+WN	0.14330

TABLE 1: Results of four kernels used in one-dimensional regression problem. The first two kernels are simply constructed kernels and the third kernel is structure-search-generated kernel, and the last kernel is SS-generated kernel+white noise. These results are yielded after repeatedly adjust the parameters of the model and minimize steps et.al.

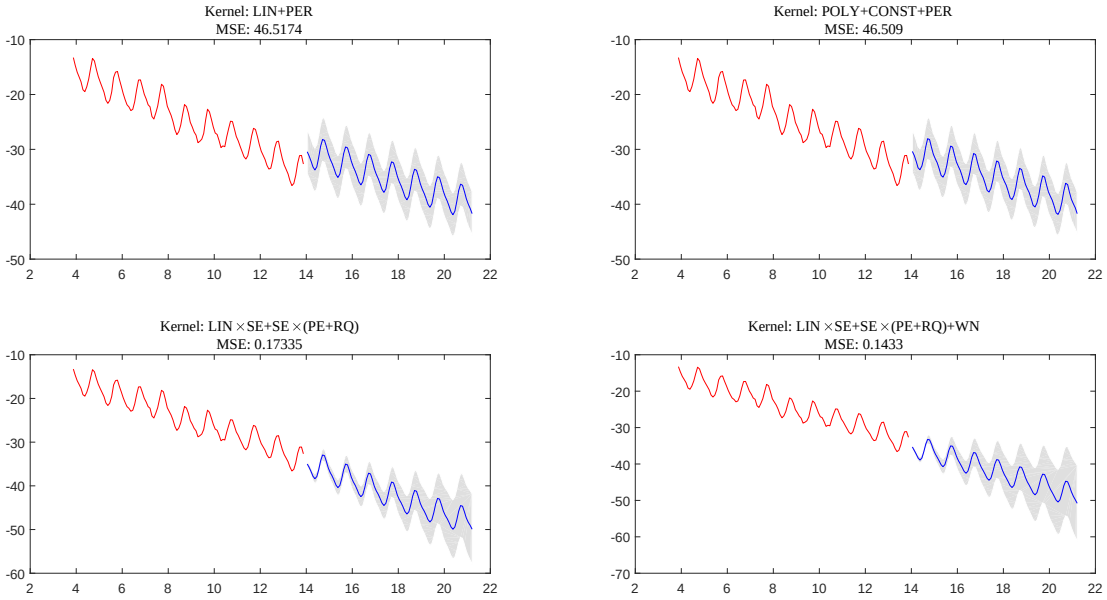

Fig. 2: Regression performance of four kernels on the one-dimensional data. The red part corresponds to train data and the blue part corresponds to test data. The gray region show the regions with confidence of 95%.

MO	N_{train}	MSE
5	50	0.0888
5	100	0.0682
5	300	0.0508
5	1000	0.0437
1	50	0.0431
1	100	0.0469
1	300	0.0354
1	**1000**	**0.0317**

TABLE 2: Regression result of additive kernel with different maximum order and train dataset size. This table lists only a part of experiment results. In simulation the maximum order decreases one per step from 40 to 1.

5 Experimental Results

This section gives all the numerical results when selecting models. All the experiments in this article is all simulated with MATLAB R2016b and GPML toolbox. As the warped Gaussian process is implemented as a likelihood function in GPML toolbox, the results of WGP will be shown by comparing results of different likelihood functions.

5.1 One-dimensional Regression

As mentioned above, I tried four kernel functions, as Table 1 shows. And Figure 2 shows the regression performance of these kernels. Obviously the structure-search-generated kernel function have much better performance on the test data. What's more, simply adding a White Noise kernel can significantly improve the regression performance.

The final model exhibits plausible extrapolation and interpretable components: a long-term trend, annual periodicity, and medium-term deviations. The LIN×SE corresponds to long-term trend, SE×PER correspons to periodic, SE×RQ represents medium-term deviation and WN stands for noise component.

5.2 Multi-dimensional Regression

The provided data have 10000 points. If using all data to train GP, this train procedure will become extremely slow. So I pick out the first N_{train} data to train GP model. Of course, the value of N_{train} should be determined by simulation. Since this problem is a multi-dimensional problem, I first tried the additive kernel, which is specially designed for multi-dimensional problem. After repeated simulation tests, the maximum order (MO) is eventually determined to be 1. Table 2 shows part of

N_{train}	L_{train}	MSE
50	100	0.1644
100	100	0.1639
300	100	0.0310
1000	100	0.0280
1000	**200**	**0.0270**
1000	300	0.0272

TABLE 3: Regression result of SE kernel with ARD distance meature.

lik/inf	infLaplace	infEP	infVB	infMCMC	infGaussLik
likGauss	0.0354(1.0830s)	0.0354(1.1380s)	0.0354(2.4810s)	0.0559(10.752s)	0.0354(1.1270s)
likLaplace	–	0.0353(1.3560s)	0.0342(10.140s)	0.0817(10.966s)	–
likLogistic	0.1782(1.1460s)	0.1781(1.2730s)	0.1782(2.5290s)	0.1738(19.275s)	–
likSech2	0.0350(1.1960s)	0.0367(1.4380s)	0.0347(5.3440s)	0.0675(25.460s)	–
likT	0.0355(1.1060s)	–	0.0354(3.2030s)	0.0709(20.933s)	–
likGaussWarp	0.0299(1.3050s)	0.0310(1.4660s)	0.0337(2.7020s)	–	–

TABLE 4: Results of additive kernel. Values in the table follows this format: MSE (Prediction time cost). The – of likGaussWarp with infMCMC means that the training time cost is too large, and the other – means that this combination is not allowed. All these results are generated after adjusting initial hyperparameters and minimize steps repeatedly. The blue results are better than likGauss with infGaussLik and the red result is the best result.

lik/inf	infLaplace	infEP	infVB	infMCMC	infGaussLik
likGauss	0.0325(0.0737s)	0.0333(0.1315s)	0.0299(0.0796s)	0.9269(10.268s)	0.0310(0.0448s)
likLaplace	–	0.0303(0.3400s)	0.1679(0.1179s)	0.9269(10.379s)	–
likLogistic	0.1782(0.0723s)	0.1760(0.2285s)	0.1782(0.0903s)	0.9270(18.691s)	–
likSech2	0.1625(0.0664s)	0.1625(0.1865s)	0.1626(0.1041s)	0.9269(24.833s)	–
likT	0.0332(0.0717s)	–	0.0315(0.1043s)	0.9270(19.764s)	–
likGaussWarp	0.0294(0.1746s)	0.0302(0.4171s)	0.1628(0.2054s)	–	–

TABLE 5: Results of SE kernel with ARD distance measure. Values in the table follows this format: MSE (Prediction time cost). The – of likGaussWarp with infMCMC means that the training time cost is too large, and the other – means that this combination is not allowed. All these results are generated after adjusting initial hyperparameters and minimize steps repeatedly. The blue results are better than likGauss with infGaussLik and the red result is the best result.

the experiment results. The minimize step of all the following experiments are uniformly set to 100. As experimental results shows, simple structure of additive kernel have better results than complicated structure. Also MSE goes down with N_{train} gets larger. Considering the time cost of training procedure and prediction, I finally choose $N_{train} = 1000$.

Since the additive kernel with maximum order 1 is equivalent to Squared Exponential kernel, so I tried the SE kernel with Automatic Relevance Determination (ARD) distance measure. Different from the additive kernel, the effect of minimize step L_{train} cannot be neglected. Table 3 shows experimental results of SE kernel. With Automatic Relevance Determination (ARD) distance measure, the performance of GP model is better than additive kernel. After testing these two kernels, we finally apply the SE kernel with Automatic Relevance Determination distance measure.

After determining the kernel function, we have to select likelihood function and inference method because the dataset is large and multi-dimensional. In this article we choose six likelihood functions: Gaussian (likGauss), Laplacian (likLaplace), Logistic (likLogistic), Sech-square (likSech2), Student's T (likT) and Warped Gaussian (likWarpGauss)likelihood function. Also we choose four inference methods: Laplace approximation (infLaplace), Expectation Propagation approximation (infEP), Variational Bayesian approximation (infVB), Markov Chain Monte Carlo (MCMC) sampling (infMCMC). In order to compare the results, I also simulated the Gaussian inference method (infGaussLik) with Gaussian likelihood function, which gives an exact inference. Table 4 and Table 5 gives the compare result of the combinations of likelihood functions and inference methods respectively corresponds to additive kernel and SE kernel with ARD distance meature. The warp function of WGP with ARD kernel is set to $f(t, \mathbf{\Psi}) = t + \sum_{i=1}^{3} a_i \tanh\left(b_i(t + c_i)\right)$.

Combining all the above methods together and adjusting the parameters, I got the final result of the regression on two problems: the first problem MSE equals to 0.14243 and the second problem MSE equals to 0.026362.

6 Conclusion

Gaussian Process Regression is a popular regression method in recent years which yields impressive performance. In practical application, model selection is of great importance, including selections of kernel function, likelihood function and inference method. For one-dimensional regression, I implemented the structure search algorithm and compared the result kernel function with simply constructed kernels. Also I add a *white noise kernel* on the result kernel and improve the regression performance. The final best mean-square error equals to 0.14243. For multi-dimensional regression, I compared the performance of three different models: general Gaussian process, additive Gaussian process, and warped Gaussian process. In addition, I compared the effect of different combinations of

likelihood function and inference methods. Experimental results shows that *warped Gaussian process model with SE-ARD kernel inferred with Laplace approximation* have the best performance. The final best mean-square error equals to 0.026362.

Appendix A
Kernel Definitions

For scalar-valued inputs, the squared exponential (SE), periodic (PER), linear (LIN), rational quadratic (RQ), and white noise (WN) kernels are defined as follows:

$$\begin{aligned} K_{se}(x,x') &= \sigma^2 \exp\left(-\frac{(x-x')^2}{2l^2}\right) \\ K_{per}(x,x') &= \sigma^2 \exp\left(-\frac{2\sin^2(\pi(x-x')/p)}{l^2}\right) \\ K_{lin}(x,x') &= \sigma_b^2 + \sigma_v^2(x-l)(x'-l) \\ K_{rq}(x,x') &= \sigma^2\left(1+\frac{(x-x')^2}{2\alpha l^2}\right) \\ K_{wn}(x,x') &= \delta(x-x') \end{aligned} \tag{A.1}$$

Appendix B
Gaussian Conditionals

A standard result shows how to condition on a subset of dimensions $\mathbf{y}_B$ of a vector $\mathbf{y}$ having a multivariate Gaussian distribution. If

$$y = \begin{bmatrix}\mathbf{y}_A \\ \mathbf{y}_B\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{AA} & \boldsymbol{\Sigma}_{AB} \\ \boldsymbol{\Sigma}_{BA} & \boldsymbol{\Sigma}_{BB}\end{bmatrix}\right) \tag{A.2}$$

then

$$\begin{aligned} \mathbb{E}\{\mathbf{y}_A|\mathbf{y}_B\} &= \boldsymbol{\mu}_A + \boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_{BB}^{-1}(\mathbf{x}_B - \boldsymbol{\mu}_B) \\ \mathrm{cov}_{\mathbf{y}_A|\mathbf{y}_B} &= \boldsymbol{\Sigma}_{AA} - \boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_{BB}^{-1}\boldsymbol{\Sigma}_{BA} \end{aligned} \tag{A.3}$$

This result can be used in the context of Gaussian process regression, where $\mathbf{y}_B = [f(\mathrm{x}_1), f(\mathrm{x}2), \ldots, f(\mathrm{x}_N)]$ represents a set of function values observed at some subset of locations $[\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_N]$, while $\mathbf{y}_A = [f(\mathrm{x}_1^*), f(\mathrm{x}_2^*), \ldots, f(\mathrm{x}_N^*)]$ represents test points whose predictive distribution we would like to know.

Acknowledgment

I would like to thank Liren Yu, Sihan Zeng, Yuanxin Zhang, Ruoyang Liu, and Bowen Xia for their help. I had discussions about kernel selection problems with Liren Yu, Sihan Zeng and Yuanxin Zhang and I had discussions about dataset with Ruoyang Liu and Bowen Xia. These discussions help me clarify ideas and identify specific optimization goals, and also help me avoid many difficulties.

References

[1] Chalupka, K., Williams, C.K.I., Murray, I.: A framework for evaluating approximation methods for Gaussian process regression. JMLR 14(1). 2013

[2] David Kristjanson Duvenaud. Automatic Model Construction with Gaussian Processes. PhD Thesis. University of Cambridge. 2014

[3] Duvenaud D, Lloyd J R, Grosse R, et al. Structure Discovery in Nonparametric Regression through Compositional Kernel Search. Creative Commons Attribution-Noncommercial-Share Alike, 2013.

[4] Duvenaud, D., Nickisch, H., and Rasmussen, C.E. Additive Gaussian processes. In Advances in Neural Information Processing Systems, 2011.

[5] E. Snelson, Z. Ghahramani, and C. Rasmussen. Warped Gaussian processes. In Advances in Neural Information Processing Systems 16, 2003.

[6] Ferrari Trecate, G., Williams, C. K. I., and Opper, M. (1999). Finite-dimensional Approximation of Gaussian Processes. In Kearns, M. S., Solla, S. A., and Cohn, D. A., editors, Advances in Neural Information Processing Systems 11, pages 218¨C224. MIT Press.

[7] Francis Bach. High-dimensional non-linear variable selection through hierarchical kernel learning. CoRR, abs/0909.0844, 2009.

[8] Gibbs, M. N. (1997). Bayesian Gaussian Processes for Regression and Classification. PhD thesis, Department of Physics, University of Cambridge.

[9] Grosse, R.B., Salakhutdinov, R., Freeman, W.T., and Tenenbaum, J.B. Exploiting compositionality to explore a large space of model structures. In Uncertainty in Artificial Intelligence, 2012.

[10] Hastie, T.J. and Tibshirani, R.J. Generalized additive models. Chapman & Hall/CRC, 1990.

[11] I.G. Macdonald. Symmetric functions and Hall polynomials. Oxford University Press, USA, 1998.

[12] Lloyd J R, Duvenaud D, Grosse R, et al. Automatic construction and natural-language description of nonparametric regression models. Twenty-Eighth AAAI Conference on Artificial Intelligence. AAAI Press, 2014.

[13] Miguel Lázaro-Gredilla. Bayesian Warped Gaussian Processes. Advances in Neural Information Processing Systems 25. 2012

[14] Rasmussen, C.E. and Williams, C.K.I. Gaussian Processes for Machine Learning. The MIT Press, Cambridge, MA, USA, 2006.

[15] Rastogi A. Theory and Algorithms for Modern Problems in Machine Learning and an Analysis of Markets. PhD Thesis. New York University. 2008

[16] R.P. Stanley. Enumerative combinatorics. Cambridge University Press, 2001.

[17] Schwarz, G. Estimating the dimension of a model. The Annals of Statistics, 6(2):461-464, 1978.

[18] T.P. Minka. Expectation propagation for approximate Bayesian inference. In Uncertainty in Artificial Intelligence, volume 17, pages 362¨C369, 2001.

[19] Wilson G. Andrew, Adams P. Ryan. Gaussian process kernels for pattern discovery and extrapolation. International Conference on Machine Learning. 2013.

[20] Y. Bengio, O. Delalleau, and N. Le Roux. The curse of highly variable functions for local kernel machines. Advances in neural information processing systems, 18, 2006.