May 2016

Gaussian Process Regression for Large Data Sets

Nicolas Kuhaupt
University of Wisconsin-Milwaukee

GAUSSIAN PROCESS REGRESSION FOR LARGE DATA SETS

by

Nicolas Kuhaupt

A Thesis Submtted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Mathematics

at

The University of Wisconsin-Milwaukee

May 2016

ABSTRACT

GAUSSIAN PROCESS REGRESSION FOR LARGE DATA SETS

by

Nicolas Kuhaupt

The University of Wisconsin-Milwaukee, 2016
Under the Supervision of Professor Jugal Ghorai

Gaussian Process Regression is a non parametric approach for estimating relationships in data sets. For large data sets least square estimates are not feasible because of the covariance matrix inversion which requires $O(n^3)$ computation. In Gaussian Process Regression a matrix inversion is also needed, but approximation methods exists for large n. Some of those approaches are studied in this thesis, among them are the random projection of the covariance matrix, Nyström method and the Johnson-Lindenstrauß Theorem. Furthermore sampling methods for Hyperparameter estimation are explored.

Table of contents

List of figures

ACKNOWLEDGEMENTS

First I would like to acknowledge my supervisor Prof. Dr. Jugal Ghorai. He lay the basis for my thesis, prepared a lot of theory in his seminar and has a style of explaining concepts intuitively which helped understanding a lot.

Furthermore I would like to acknowledge Prof. Dr. Anjishnu Banerjee whose Thesis was also an inspiration for me and motivated the problems I was working on.

Thanks go to the Math Department of University of Wisconsin - Milwaukee and University of Ulm. There are to many people who made my studies a great experience to name them all.

I want to acknowledge my family back home in Germany who I left behind for one year but nonetheless supported me and who made all that possible for me. My deepest owing to all of them!

Last I would like to acknowledge my friends and roommates who grow dear to my heart and have been a family to me in the last 10 months.

1 Introduction

Data is growing rapidly in all areas. Not only online data is doubling every year but also new technologies like genome sequencing contribute to a rapid growth [1]. It is estimated, that 80% of data is unstructured. Recent algorithms in artificial intelligence are nonetheless able to make sense out of unstructured data which makes those data also prone to statistical methods. Furthermore algorithms also tend to produce data. For example IBM´s Watson, one of the leading artificial intelligence algorithms has opened an API which is dedicated to recognize emotions in all kind of media. In consequence out of every media new data can arise. And to name one last recent development among others: The internet of things. Nearly every new technology is connected to the internet, from cars to refrigerators and health care technologies. Measurements are therefore cheap and easy and produce tons of data in consequence. On the other side technology also improves computational capacity which results in faster and cheaper computation. Moore predicted in 1965, that every two years the number of transistors per chip double[2]. This is known as Moore´s Law and until today his prediction is closely matched. However, if we assume a doubling rate on both side, computation and data growth, the missing variable are the algorithms which use

the computation capacity to make sense out of the data. Statistical methods like least square estimation need $O(n^3)$ time. So the growth of computation capacity does not keep up with the data growth, if we want to apply the precise techniques we have. One refers to this as Big Data problem or simply big n problem. Workarounds and approximations are needed which build the field of machine learning. This is the motivation for this thesis. Here one technique among many is explored: The Gaussian Process Regression. Computationally $O(n^3)$ is also needed for an inversion of the covariance matrix but many different approximation methods exists which reduce it to an inversion of lower rank.

The procedure in this thesis is roughly the following: We will introduce the theory of Gaussian Process Regression and replace the Gaussian Process based on n observations by a Process based on m observations with $m << n$. This makes the process smoother and enables us together with the presented Woodbury matrix identity to invert only an $m \times m$ matrix instead of $n \times n$. Next we face approximation methods which make the computations efficient. The approximation methods we present here are: Rank reduction by random projection, the nyström method and an algorithm which combines the Johnson Lindenstrauß Theorem and the idea of the nyström method. Last, we

assume a certain covariance function for the Gaussian Process and want to learn the parameters from the given Data. Two methods will be presented: The maximum likelihood maximization and a Bayesian approach where we sample the parameters from their priori distribution. The needed sampling methods will be introduced.

Among those methods we implemented the random projection method for rank reduction and the marginal likelihood maximization for estimating the hyperparameters and tested it on a real data set from a robot arm.

Chapter 2 is dedicated to the theory and mathematical background. First in 2.1 the basics of Gaussian Process Regression are introduced, in 2.2 the big n problem will be faced by the 3 different approaches. In 2.3 the techniques to estimate the parameters of the covariance function are described. Here three important and widely applicable algorithms are stated and intuitive reasoning for their steps is given: Acceptance and rejection method, Gibbs sampling and the Metropolis Hastings algorithm. In chapter 3 an example application is given with a random projection and a data set of a robot arm SARCOS. We end with a conclusion in chapter 4.

2 Theory

2.1 Gaussian Process Regression

For a Gaussian Process we assume that every finite subset of , say n observations has a multivariate normal distribution $\sim N(\vec{\mu}, K_{n\times n})$ where $\vec{\mu}$ is a vector of n means and $K_{n\times n}$ a $n\times n$ covariance matrix. Therefore every single observation is normal distributed $\sim N(\mu, \sigma^2)$ for some mean μ and variance σ^2. The next step to a Gaussian Process in a continuous space is to define the normal distribution over functions. The Gaussian Process is therefore fully specified by two functions, the mean function $m(x)$ and the covariance function or covariance kernel $K(x, x)$ and we write $\sim N(m(x), K(x, x))$. For Gaussian Process Regression we want to learn about the parameters of the two functions from the given data. This enables us to understand the data better and make predictions at unobserved points. Often the mean functions is a priori chosen to be 0, which does not mean that the posteriori mean is 0 in general. However, other settings are possible, for example we can set a fixed mean function or a set of basis functions. We then want to learn about the weights for the basis functions. In the later case we would have a linear model as a mean function and the Gaussian Process models the residuals.

In practice this is used to incorporate prior knowledge about the data or because of interpretation purposes. Furthermore we have to assume a certain structure of the covariance function whose parameters we want to learn.
A widely used covariance function which will be of further interest for us is the squared exponential function:

$$cov(x, y) = \frac{1}{\theta_1} * exp\Big[-\frac{(x-y)^2}{\theta_2}\Big] \tag{1}$$

We call the parameters θ_1 , θ_2 hyperparameter. Later a noise variance σ^2 will be added. Those are the parameters we want to learn from the data. Let´s make a few observations on the squared exponential covariance function:

1. In general we expect in a regression context that points which are close to each other are also more informative. In Gaussian Process settings the covariance function takes care of this. Note that for $x \to y$ we have $cov(x, y) \to \frac{1}{\theta_1}$

2. We call a covariance function stationary if it depends on $|x - y|$ which is the case in the squared exponential covariance function. This results in translation invariance: the covariance depends only on the distance between the observations.

3. In general we need the covariance function to fulfill $cov(x, y) = cov(y, x)$ respectively to be symmetric. Observe that this is matched in the case of squared exponential covariance function. Furthermore the squared exponential function is not only a continuous one (as it is composed of continuous functions) but also very smooth [3].

Now we will make some general observations about Gaussian Processes which we will need throughout the thesis. Let $(t_1, ..., t_n)$ be the points at which we make the observations and $Y(t)$ the corresponding observed value. Now we can model $Y(t)$ as:

$$Y(t) = w(t) + \epsilon(t) \tag{2}$$

where $w(t)$ models the Gaussian Process and we have $w(t) \sim N(0, K_{n\times n})$ for some $n \times n$ covariance matrix $K_{n\times n}$ and the noise $\epsilon \sim N(0, \sigma^2)$. We will now modify w(t) in 3 steps:

First: At the moment we incorporate all n observations. This makes predictions computationally difficult for big n as it will include inversion of a $n \times n$ matrix. Therefore we will choose a subset of observations at $(t_1^*...t_m^*)$ from $(t_1, ..., t_n)$ with $m << n$ and replace w(t) by $\mathbb{E}[w(t)|(w(t_1^*)...w(t_m^*))]$. $(t_1^*...t_m^*)$

are also called knots. We will discuss the choice of the knots later in section 2.2.1. Some notation first:

$K_{n\times n}$ is a $n \times n$ matrix with $(i,j)^{th}$ element $K(t_i, t_j)$

$K^*_{m\times m}$ is the covariance matrix of the knots with $(i,j)^{th}$ element $K(t^*_i, t^*_j)$

$K^*_{1\times m}(t)$ is the covariance of t with $(t^*_1 ... t^*_m)$.

$K^+_{n\times m}$ is the covariance matrix with $(i,j)^{th}$ element $K(t_i, t^*_j)$

$\vec{w}^* = (w(t^*_1), ..., w(t^*_m))^T$

$\vec{w} = (w(t_1), ..., w(t_n))^T$

Then we observe that

$$\begin{pmatrix} \vec{w} \\ \vec{w}^* \end{pmatrix} \sim N\left(\vec{0}, \begin{pmatrix} K_{n\times n} & K^+_{n\times m} \\ {K^+_{n\times m}}^T & K^*_{m\times m} \end{pmatrix}\right)$$

and (by Appendix Lemma 8) we have that

$$w(t)|\vec{w}^* \sim N\big(K^*_{1\times m}(t)(K^*_{m\times m})^{-1}\vec{w}^*, K(t,t) - K^*_{1\times m}(t)(K^*_{m\times m})^{-1}K^*_{m\times 1}(t)\big)$$

Second: Replace $w(t)$ by

$$\tilde{w}(t) = \mathbb{E}[w(t)|\vec{w}^*] = K^*_{1\times m}(t)(K^*_{m\times m})^{-1}\vec{w}^*$$

Then

$$\mathbb{E}(\tilde{w}(t)) = \mathbb{E}\big[\mathbb{E}(w(t)|\vec{w}^*)\big] = \mathbb{E}(w(t)) = 0$$

and

$$\begin{aligned} Var(\tilde{w}(t)) &= Var(K^*_{1\times m}(t)(K^*_{m\times m})^{-1}\vec{w}^*) \\ &= K^*_{1\times m}(t)(K^*_{m\times m})^{-1}Var(\vec{w}^*)(K^*_{1\times m}(t)(K^*_{m\times m})^{-1})^T \\ &= K^*_{1\times m}(t)(K^*_{m\times m})^{-1}(K^*_{m\times m})(K^*_{m\times m})^{-1}K^*_{m\times 1}(t) \\ &= K^*_{1\times m}(t)(K^*_{m\times m})^{-1}K^*_{m\times 1}(t) \end{aligned}$$

In short:

$$\tilde{w}(t) \sim N(0, K^*_{1\times m}(t)(K^*_{m\times m})^{-1}K^*_{m\times 1}(t)) \tag{3}$$

And we have $Y(t) = \tilde{w}(t) + \epsilon(t)$

Third: We observe that the variance of the process $\tilde{w}(t)$ underestimates the variance of $w(t)$ as

$$Var(\tilde{w}(t)) = K^*_{1\times m}(t)(K^*_{m\times m})^{-1}K^*_{m\times 1}(t) < K(t,t) = Var(w(t)) \tag{4}$$

since

$$Var(w(t)|\vec{w}^*) = K(t,t) - K^*_{1\times m}(t)(K^*_{m\times m})^{-1}K^*_{m\times 1}(t) > 0 \tag{5}$$

To remedy the underestimation we add an extra noise $\tilde{\epsilon}$ [4] with

$$cov(\tilde{\epsilon}(s), \tilde{\epsilon}(t)) = K(s,t) - K^*_{1\times m}(s)(K^*_{m\times m})^{-1}K^*_{m\times 1}(t) \tag{6}$$

to obtain finally

$$Y(t) = \tilde{w}(t) + \tilde{\epsilon}(t) + \epsilon(t) \tag{7}$$

and

$$cov(Y(s), Y(t)) = \tilde{K}(s,t) + [K(s,t) - \tilde{K}(s,t)]I(s,t) + \sigma^2 I(s,t) \tag{8}$$

with $\tilde{K}(s,t) = K^*_{1\times m}(s)(K^*_{m\times m})^{-1}K^*_{m\times 1}(t)$.

Note 1

The constructed covariance matrix has to be non degenerate to be invertible. To ensure this we show that the covariance matrix is positive definite

respectively for every $n \times 1$ vector $\vec{u}$ of non-zero real numbers we have

$$\vec{u}^T_{1\times n} cov(Y(t_1), ..., Y(t_n))\vec{u}_{n\times 1} > 0$$

Proof.

$$\vec{u}^T_{1\times n} cov(Y(t_1), ..., Y(t_n))\vec{u}_{n\times 1}$$

$$= \vec{u}^T_{1\times n}\big[K^+_{n\times m}(K^*_{m\times m})^{-1}K^+_{m\times n} + (K_{n\times n} - K^+_{n\times m}(K^*_{m\times m})^{-1}K^+_{m\times n})I_{n\times n} + \sigma^2 I_{n\times n}\big]\vec{u}_{n\times 1}$$

$$= \vec{u}^T_{1\times n}K^+_{n\times m}(K^*_{m\times m})^{-1}K^+_{m\times n}\vec{u}_{n\times 1} +$$

$$\vec{u}^T_{1\times n}\big[(K_{n\times n} - K^+_{n\times m}(K^*_{m\times m})^{-1}K^+_{m\times n})I_{n\times n} + \sigma^2 I_{n\times n}\big]\vec{u}_{n\times 1}$$

Further we define $\vec{v}^T_{1\times m} = \vec{u}^T_{1\times n}K^+_{n\times m}$ and therefore

$$\vec{u}^T_{1\times n}K^+_{n\times m}(K^*_{m\times m})^{-1}K^+_{m\times n}\vec{u}_{n\times 1} = \vec{v}^T_{1\times m}(K^*_{m\times m})^{-1}\vec{v}_{m\times 1} \geq 0$$

since $K^*_{m\times m}$ is a covariance matrix and therefore positive-semidefinite by definition. Observing that $\sigma^2 > 0$ and from (5) that $(K_{n\times n} - K^+_{n\times m}(K^*_{m\times m})^{-1}K^+_{m\times n})_{ii} > 0$ for $i = 1, ..., n$ finishes the proof. □

Note 2

For predictions at an unobserved point t_0 we use the following:

$$\hat{Y}(t_0) = K_{1\times n}(t_0)(K_{n\times n} + \sigma^2 I_{n\times n})^{-1}\vec{Y}_{n\times 1} \tag{9}$$

Observe that this is a linear combination of the observations $Y(t_1), ..., Y(t_n)$.

2.2 Rank Reduction

In the following section we will face the "big n" problem in Gaussian Process Regression through three different approaches. In section 2.2.1 we will take a subset of observations which will be chosen through a random projection. In the next section we will exploit three important Theorems to rewrite the covariance kernel and the Gaussian Process in terms of a series of its eigenvalues and eigenfunctions and truncate this series. Last in section 2.2.3 we will use matrix approximation techniques based on Johnson-Lindenstrauß' Theorem. However, in all sections we will make use of the Woodbury Matrix Identity, which allows to reduce the calculation of the inverse of a $n \times n$ matrix to the inversion a $m \times m$ matrix with $m < n$ if the $n \times n$ matrix has a certain structure.

Theorem 1 (Woodbury Matrix Identity)

$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$

for $A \in \mathbb{R}^{n \times n}, U \in \mathbb{R}^{n \times m}, C \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{m \times n}$ and $m < n$.

Proof. A fast forward way is by multiplication of $(A + UCV)$ on both sides and derive the Identity matrix from the right hand side. However, we follow a different approach by blockwise elimination. Let:

$$\begin{bmatrix} A & U \\ V & -C^{-1} \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix}$$

Matrix multiplication leads to

$$AX + UY = I \tag{10}$$

$$VX - C^{-1}Y = 0 \tag{11}$$

Solving (11) for Y and plug into (10) leads to

$$AX + UCVX = I$$

$$X = (A + UCV)^{-1} \tag{12}$$

From (10) we can also get: $X = A^{-1}(I - UY)$ which we plug into (11) to

get:

$$VA^{-1}(I - UY) = C^{-1}Y$$

$$VA^{-1} = C^{-1}Y + VA^{-1}UY$$

$$VA^{-1} = (C^{-1} + VA^{-1}U)Y$$

$$Y = (C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

plugging Y back into (10) gives:

$$AX + U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} = I$$

$$X = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

This together with (12) finishes the proof. □

Note 3

In our context the matrix A will be a diagonal matrix and therefore the inverse of A is just the reciprocal of its diagonal elements. Further observe that on the right hand side of the Woodbury Matrix Identity we need to invert a $m \times m$ matrix instead of $n \times n$ on the left hand side.

2.2.1 Random Projection

Let $\phi_{m\times n}$ be a random permutation of $I_{n\times n}$ which cuts of after row m. We can obtain the knots in the following way:

$$\begin{pmatrix} t_1^* \\ \vdots \\ t_m^* \end{pmatrix} = \phi_{m\times n} \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix}$$

and

$$\begin{pmatrix} w(t_1^*) \\ \vdots \\ w(t_m^*) \end{pmatrix} = \phi_{m\times n} \begin{pmatrix} w(t_1) \\ \vdots \\ w(t_n) \end{pmatrix}$$

As developed in section 2.1 we now have:

$$Y \sim N(0, \tilde{K}(s,t) + [K(s,t) - \tilde{K}(s,t)]I(s,t) + \sigma^2 I(s,t))$$

with

$$\tilde{K}(s,t) = K^*_{1\times m}(s)(K^*_{m\times m})^{-1}K^*_{m\times 1}(t).$$

The advantages of this approach are obvious: It is simple to implement and computationally undemanding. Furthermore in [5] this approach shows

better accuracy in both simulated and real data prediction in comparison to more sophisticated knot choices for example equidistant knots, which require further computation.

2.2.2 Nyström method

The goal in this section is to rewrite the Gaussian Process $w(t)$ and the covariance kernel $K(s,t)$ as a series of its eigenfunctions and eigenvalues and cut them off after the m^{th} summand with $m << n$ which will give us an approximation where we can apply the Woodbury matrix identity. In order to do so we need a little theory:

Definition 1 (Eigenfunction)

Let $K(s,t) = cov(s,t)$ be a covariance function. Then we call $\phi_1, \phi_2, ..., \phi_n$ eigenfunctions of K with corresponding eigenvalues $\lambda_1, ..., \lambda_n$ if it satisfies the following integral equation:

$$\int K(s,t)\phi_i(s)ds = \lambda_i\phi_i(t) \tag{13}$$

Theorem 2 (Mercer's Theorem)

Let K be a continuous Kernel and $T = [a; b] \subset \mathbb{R}$, then T_K has an orthonormal basis of eigenfunctions $\{\psi_i\}_{i\in\mathbb{N}}$ and corresponding eigenvalues $\{\lambda_i\}_{i\in\mathbb{N}}, \lambda_i > 0\ \forall i \in \mathbb{N}$ in $L^2(T, \mathcal{B}_T, v)$, and

$$K(s,t) = \sum_{j=1}^{\infty} \lambda_j \psi_j(s)\psi_j(t) \quad s,t \in T$$

where this convergence is absolute and uniform on T.

Proof. Without proof. □

Note 4

The assumption that the kernel is continuous is generally fulfilled for covariance functions. Especially for the squared exponential covariance function as defined in (1) we see that it is a composition of continuous functions and therefore continuous.

Mercer's Theorem therefore allows us to rewrite the covariance matrix as a series of its eigenvalues and eigenfunctions. Next Karhunen-Loève Expansion allows us to write the process $w(t)$ in terms of its eigenfunctions and eigenvalues:

Theorem 3 (Karhunen-Loève Expansion)

For any centered $X = \{X(t), t \in T\}$ on a compact T with continuous co-

variance function K, there exists a family $\{\xi\}_{n=1}^{\infty}$ of uncorrelated random variables with $\mathbb{E}\xi_n = 0$ and $\mathbb{E}\xi_n^2 = 1$ such that

$$X(t) = \sum_{n=1}^{\infty} \sqrt{\lambda_n}\xi_n\psi_n(t) \tag{14}$$

where $\{\psi\}_{n=1}^{\infty}$ is an orthonormal basis of eigenfunctions, λ_n are the eigenvalues corresponding to ψ_n and the above expansion holds in $L^2(\Omega, \mathcal{F}, \mathcal{P})$.

Proof. Without proof. □

Note 5

The process X needs to be centered in Karhunen-Loève Theorem, i.e. $\mathbb{E}(X) = 0$ in T. This is fulfilled for $w(t)$.

This allows us to rewrite the process $w(t)$ as $w(t) = \sum_{j=1}^{\infty} \sqrt{\lambda_j} e_j(t)\xi_j$ with ξ_j iid $\sim N(0, 1)$.

Theorem 4 (Eckart-Young-Mirsky Theorem)

Let $A = U_n\Sigma_n V_n^T$ be a singular value decomposition of A with U and V unitary matrices and Σ diagonal matrix with diagonal elements $(\sigma_1, ...\sigma_n)$ such that $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$. Then the best rank k approximation to A in

terms of the Frobenius norm is given by

$$A^k = \sum_{i=1}^{k} u_i \sigma_i v_i^t \tag{15}$$

where u_i and v_i are the i^{th} row of U and V respectively.

To prove the Eckart-Young-Mirsky Theorem we need the following

Lemma 1

For the Frobenius norm of a matrix $A \in \mathbb{R}^{n \times m}$ it holds that:

$$\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{i,j}|^2} = \sqrt{Tr(AA^T)} = \sqrt{Tr(A^T A)} \tag{16}$$

with $Tr(AA^T)$ the trace of AA^T.

Proof of Lemma. To see that this equality holds examine the diagonal entries of AA^T:

$$(AA^T)_{1,1} = a_{1,1}^2 + a_{1,2}^2 + a_{1,3}^2 + ... + a_{1,m}^2$$

$$(AA^T)_{2,2} = a_{2,1}^2 + a_{2,2}^2 + a_{2,3}^2 + ... + a_{2,m}^2$$

$$\vdots$$

$$(AA^T)_{n,n} = a_{n,1}^2 + a_{n,2}^2 + a_{n,3}^2 + ... + a_{n,n}^2$$

and we see that summation over the diagonal elements gives the Frobenius norm. □

Proof of Theorem. Let B be a matrix of the same dimensions as A. We will show that B has the form of A^k in order to minimize $\|A - B\|_F$. We define $D = U^T B V$. $\implies B = UDV^T$

Hence

$$\|A - B\|_F^2 = \|U\Sigma V^T - UDV^T\|_F^2 = \|U(\Sigma - D)V^T\|_F^2$$

$$\underset{Lemma\ 1}{=} Tr(U(\Sigma - D)V^T(U(\Sigma - D)V^T)^T)$$

$$= Tr(U(\Sigma - D)V^TV(\Sigma - D)^TU^T) \underset{Vunitary}{=} Tr(U(\Sigma - D)(\Sigma - D)^TU^T)$$

$$= Tr(U(\Sigma - D)(U(\Sigma - D))^T) = Tr((U(\Sigma - D))^TU(\Sigma - D))$$

$$= Tr((\Sigma - D)^TU^TU(\Sigma - D)) = Tr((\Sigma - D)^T(\Sigma - D))$$

$$= \|\Sigma - D\|_F^2 \underset{\Sigma \text{ diagonal matrix}}{=} \sum_i |\sigma_i - D_{ii}|^2 + \sum_{i \neq j} |D_{ij}|^2$$

$$= \sum_{i \leq k} |\sigma_i - D_{ii}|^2 + \sum_{i > k} |\sigma_i - D_{ii}|^2 + \sum_{i \neq j} |D_{ij}|^2$$

and we observe that for D of rank k this is minimized by $D_{ii} = \sigma_i$ for $i \leq k$ and $D_{ij} = 0$ for $i \neq j$ and hence $B = UDV^T$ matches A^k □

Note 6

In the Theorem of Eckart-Young-Mirsky it is made use of singular value decomposition. However, as in our setting the covariance matrix K is diagonalizable, we can make use of eigendecomposition $K = E\Lambda E^T$ where E is a $n \times m$ matrix whose columns are the eigenvectors and Λ a diagonal matrix with the eigenvalues as its diagonal elements. Now with the Eckart-Young-Mirsky Theorem it follows that the best approximation to K is by its largest eigenvalues.

Conclusion

The covariance of the approximated process is given by

$$K = E_{n\times m}\Lambda E^T_{n\times m} + \sigma^2 I_{n\times n} \tag{17}$$

(Note that we can apply the Woodbury Matrix Identity to calculate the inverse) and for predictions we can use the formula derived from the Karhunen-Loève Expansion:

$$Y(t) = \sum_{j=1}^{m} \sqrt{\lambda_j} e_j(t)\xi_j \tag{18}$$

with ξ_j iid $\sim N(0,1)$.

However, the Rank Reduction through Truncated Series Expansion is ap-

plicable if we already know the eigenvalues and eigenvectors. Otherwise computation takes time $O(n^3)$ and we have not gained anything since matrix inversion also takes $O(n^3)$. Therefore we will next derive an approximation method for eigenvalues and eigenfunctions.

The procedure to derive the approximation is roughly the following: We calculate eigenvectors and corresponding eigenvalues of a sub matrix. Those are then used to extend it to an approximation of eigenvalues and eigenvectors of the full matrix. Therefore, last, we know by Mercer's Theorem that if we have the eigenvalues and eigenvectors we have an approximation to the matrix. Let´s start off with some theory:

We want to find the eigenfunctions $\Phi()$ of the covariance kernel as defined in Definition 1:

$$\int_a^b K(x,s)\Phi_i(s)ds = \lambda_i\Phi_i(x) \tag{19}$$

(This integral equation is also known as a Fredholm integral equation of the second kind and no analytical solution is known). Nyströms method to approximate an integral is to replace the integral by a weighted sum and a

choice of knots $(s_1, ..., s_m)$ where we evaluate the corresponding functions:

$$\int_a^b K(x,s)\Phi_i(s)ds \approx \sum_{j=1}^{m} w_j k(x, s_j)\Phi_i(s_j) \tag{20}$$

Therefore we have to find approximations $\tilde{\phi}$ and $\tilde{\lambda}$ such that

$$\sum_{j=1}^{m} w_j k(x, s_j)\tilde{\phi}_i(s_j) = \tilde{\lambda}_i\tilde{\phi}_i(x) \tag{21}$$

Lets further give each point equal weight by choosing $w_j = \frac{1}{m}$ for $j = 1...m$ and choose the knots $s_j = x_j$ for $j = 1, ..., m$ (which corresponds to m observations):

$$\frac{1}{m}\sum_{j=1}^{m} k(x, x_j)\tilde{\phi}_i(x_j) = \tilde{\lambda}_i\tilde{\phi}_i(x) \tag{22}$$

Lets now assume we plug in $(x_1, ..., x_m)$ for x. Then we arrive at the matrix eigenproblem

$$K_{m\times m}U = \Lambda^{(m)}U \tag{23}$$

where $\Lambda^{(m)}$ is a $m \times m$ diagonal matrix with diagonal elements $\lambda_1^{(m)}, ..., \lambda_m^{(m)}$, K a $m \times m$ matrix with $(i,j)^{th}$ element $K(x_i, x_j)$ and U a $m \times m$ matrix and

orthonormal. If we compare (22) and (23) we arrive at an estimator for λ_i:

$$\lambda_i \approx \frac{\lambda_i^{(m)}}{m} \tag{24}$$

and for $\tilde{\phi}_i(x)$ for orthogonality ($\frac{1}{m}\sum_{k=1}^{m}\phi_i(x_k)\phi_j(x_k) = \delta_{i,j}$) we have:

$$\tilde{\phi}_i(x_j) \approx \sqrt{m}U_{j,i}^{(m)} \tag{25}$$

Now lets isolate $\phi(x)$ in 22 to get

$$\tilde{\phi}_i(x) \approx \frac{1}{m\lambda_i}\sum_{j=1}^{m} k(x, x_j)\phi_i(x_j) \tag{26}$$

and plug in 24 and 25 we get:

$$\phi_i(x) \approx \frac{\sqrt{m}}{\lambda_i^{(m)}}\sum_{j=1}^{m} k(x, x_j)U_{k,i}^{m} = \frac{\sqrt{m}}{\lambda_i^{(m)}}k_{1\times m}\vec{u}_i^{(m)} \tag{27}$$

where $k_{1\times m}$ is the vector $(K(x_1, x_1), K(x_1, x_2), ..., K(x_1, x_m))^T$.

The same technique can be applied to get an approximation to an $n \times n$ matrix by calculating only the eigenvectors and eigenvalues of a $m \times m$ sub

matrix. For example in Williams and Seeger [6] they derive the following:

$$\lambda_i^{(n)} = \frac{n}{m}\lambda_i^{(m)} \text{ for } i = 1, ..., m$$
$$\vec{u}_i^{(n)} = \sqrt{\frac{m}{n}}\frac{1}{\lambda_i^{(m)}}K_{n,m}\vec{u}_i^{(m)} \text{ for } i = 1, ..., m$$

where $\lambda_i^{(m)}$ and $\vec{u_i}^{(m)}$ are the calculated eigenvalues and eigenvectors of a $m \times m$ sub matrix and $\lambda_i^{(n)}$ and $\vec{u_i}^{(n)}$ are the approximated values for the full matrix. With the approximated eigenvalues and eigenvectors and the help of Mercers Theorem we can approximate a $n \times n$ matrix by:

$$\begin{aligned}\tilde{K}_{n\times n} &= \sum_{i=1}^{m}\lambda_i^{(n)}U_i^{(n)}(U_i^{(n)})^T \\ &= \sum_{i=1}^{m}\frac{n}{m}\lambda_i^{(m)}\sqrt{\frac{m}{n}}\frac{1}{\lambda_i^{(m)}}K_{n\times m}\vec{u}_i^{(m)}\sqrt{\frac{m}{n}}\frac{1}{\lambda_i^{(m)}}(\vec{u}_i^{(m)})^T K_{n\times m}^T\end{aligned}$$

$$\begin{aligned}&= \sum_{i=1}^{m}K_{n\times m}\vec{u}_i^{(m)}(\lambda_i^{(m)})^{-1}(\vec{u}_i^{(m)})^T K_{n\times m}^T \\ &= K_{n\times m}\underbrace{\Big[\sum_{i=1}^{m}(\lambda_i^{(m)})^{-1}\vec{u}_i^{(m)}(\vec{u}_i^{(m)})^T\Big]}_{K_{m\times m}^{-1}}K_{n\times m}^T\end{aligned}$$

Conclusion

The Nyström method allowed us to calculate only m eigenwerte and approximate the $n \times n$ matrix by $\tilde{K} = K_{n\times m} K_{m\times m}^{-1} K_{n\times m}^{T}$. Adding noise $\sigma^2 I_{n\times n}$ allows use of the Woodbury Matrix Identity to calculate the inverse.

2.2.3 Johnson-Lindenstrauß Theorem

The intuition of the Johnson-Lindenstrauß Theorem is that we project a set of points of a high dimensional space into a lower dimensional space and the distance between points experiences nearly no distortion. Therefore the mapping into a lower dimensional space obtains characteristics of a higher one. This has some useful application for the Gaussian Process as we can approximate the calculation through ones in a lower subspace and are therefore computationally less expensive. Let's state the Theorem first and prove it later:

Theorem 5 (Johnson-Lindenstrauß Theorem)

Let $0 < \epsilon < 1$ and

$$k \geq 4\Big(\frac{\epsilon^2}{2} - \frac{\epsilon^2}{3}\Big)^{-1} ln(n) \tag{28}$$

and V a set of n points $\in \mathcal{R}^d$. Then there exists a map $f : \mathcal{R}^d \mapsto \mathcal{R}^k$, such

that $\forall u, v \in V$ we have

$$(1-\epsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\epsilon)\|u-v\|^2. \tag{29}$$

Furthermore this map can be found in randomized polynomial time.

Note 7

The points u and v are here mapped from a d-dimensional space into a k-dimensional space with $k < d$.

In the proof we need the following

Lemma 2

Let $X \sim N(0,1)$. Then we have:

$$\mathbb{E}(e^{sX^2})) = \frac{1}{\sqrt{1-2s}} \tag{30}$$

Proof.

$$\mathbb{E}(e^{sX^2}) = \int_{-\infty}^{\infty} e^{sX^2} f(x)dx = \int_{-\infty}^{\infty} e^{sX^2} \frac{1}{\sqrt{2\pi}} e^{-\frac{X^2}{2}} dx$$
$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-X^2(\frac{1}{2}-s)} \underset{u=x\sqrt{\frac{1}{2}-s}}{=} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-u^2} \frac{1}{\sqrt{\frac{1}{2}-s}} du$$
$$= \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{\frac{1}{2}-s}} \underbrace{\int_{-\infty}^{\infty} e^{-u^2} du}_{Gaussian\ integral} = \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{\frac{1}{2}-s}} \sqrt{\pi} = \frac{1}{\sqrt{1-2s}}$$

□

The next Lemma gives bounds for a projection of a d-dimensional random vector onto its first k coordinates, which we will use to prove the Johnson-Lindenstrauß Theorem.

For the following let $X_1, ..., X_d \sim N(0,1), \vec{Y} = \frac{1}{\|X\|}(X_1, ..., X_d)$ and $\vec{Z} \in \mathcal{R}^k$ the projection of $\vec{Y}$ onto its first k coordinates and $L = \|Z\|^2$ with $\mu = \mathbb{E}(L) = \frac{k}{d}$

Lemma 3

For $k < d$ we have

(I) For $\beta < 1$

$$\mathbb{P}\Big(L \leq \frac{\beta k}{d}\Big) \leq \beta^{\frac{k}{2}} \Big(\frac{(1-\beta)k}{d-k}\Big)^{\frac{d-k}{2}} \leq exp\Big(\frac{k(1-\beta+ln\beta)}{2}\Big) \qquad (31)$$

and (II) for $\beta > 1$ we have:

$$\mathbb{P}\Big(L \geq \frac{\beta k}{d}\Big) \leq \beta^{\frac{k}{2}}\Big(1 + \frac{(1-\beta)k}{d-k}\Big)^{\frac{d-k}{2}} \leq exp\Big(\frac{k(1-\beta+ln\beta)}{2}\Big) \tag{32}$$

Proof. We prove part (II):

$$\mathbb{P}\Big(L \geq \frac{\beta k}{d}\Big) = \mathbb{P}\Big(\frac{X_1^2 + ... + X_k^2}{\|X\|} \geq \frac{k\beta}{d}\Big)$$

$$\mathbb{P}\big[d(X_1^2 + ... + X_k^2) \geq k\beta(X_1^2 + ... + X_d^2)\big]$$

$$= \mathbb{P}\big[d(X_1^2 + ... + X_k^2) - k\beta(X_1^2 + ... + X_d^2) \geq 0\big]$$

$$= \mathbb{P}\big[exp(d(X_1^2 + ... + X_k^2) - k\beta(X_1^2 + ... + X_d^2)) \geq 1\big]$$

since $e^x > 1$ for x positive and $e^x < 1$ for x negative, we can introduce a $t > 0$ in the exponential function and get:

$$= \mathbb{P}\Big[exp\big(t(d(X_1^2 + ... + X_k^2) - k\beta(X_1^2 + ... + X_d^2))\big) \geq 1\Big] \tag{33}$$

for the following estimation observe that if $t(d(X_1^2 + ... + X_k^2) - k\beta(X_1^2 + ... + X_d^2))$ is negative, we have that $\mathbb{P}\Big[exp\big(t(d(X_1^2+...+X_k^2)-k\beta(X_1^2+...+X_d^2))\big) \geq$

$1\Big] = 0$ and $\mathbb{E}(e^x) > 0$ for every x. For $t(d(X_1^2 + ... + X_k^2) - k\beta(X_1^2 + ... + X_d^2))$ being positive $\mathbb{P}\Big[exp\big(t(d(X_1^2 + ... + X_k^2) - k\beta(X_1^2 + ... + X_d^2))\big) \geq 1\Big] = 1$ and $\mathbb{E}(e^x) \geq 1$ for every $x \geq 1$. Hence:

$$
\begin{aligned}
&\mathbb{P}\Big[exp\big(t(d(X_1^2 + ... + X_k^2) - k\beta(X_1^2 + ... + X_d^2))\big) \geq 1\Big] \\
&\leq \mathbb{E}\Big[exp\big(t(d(X_1^2 + ... + X_k^2) - k\beta(X_1^2 + ... + X_d^2))\big)\Big] \\
&= \mathbb{E}\Big[exp\big((td - tk\beta)(X_1^2 + ... + X_k^2) - tk\beta(X_{k+1}^2 + ... + X_d^2))\big)\Big] \\
&\underset{X_i \sim N(0,1) \text{ for all } i=1,...,d}{=} \mathbb{E}\Big[exp\big((td - tk\beta)(X^2)\big)\Big]^k \mathbb{E}\Big[exp\big(- tk\beta)(X^2)\big)\Big]^{d-k} \\
&\underset{Lemma2}{=} (1 - 2(td - tk\beta)^{-\frac{k}{2}}(1 + 2tk\beta)^{-\frac{d-k}{2}} \\
&=: g(t)
\end{aligned}
$$

for $td - tk\beta < \frac{1}{2}$ and $tk\beta < \frac{1}{2}$ and from (33) $t > 0$.

$\implies 0 < t < \frac{1}{2k\beta}$

Now we want to find the smallest t, for which this holds. Solving g'(t)=0 leads to:

$$t^* = -\frac{1 - \beta}{2\beta(d - k\beta)} \tag{34}$$

And finally:

$$\mathbb{P}\Big(L \geq \frac{\beta k}{d}\Big) \leq g(t^*) = (1 - 2\frac{1-\beta}{2\beta(d-k\beta)}k\beta)^{-\frac{d-k}{2}}(1 + 2\frac{1-\beta}{2\beta(d-k\beta)}(d-k\beta))^{-\frac{k}{2}}$$
$$= \Big[1 - \frac{(1-\beta)k}{d-k\beta}\Big]^{-\frac{d-k}{2}}\Big(\frac{1}{\beta}\Big)^{-\frac{k}{2}}$$
$$= \Big[\frac{d-k\beta-k+k\beta}{d-k\beta}\Big]^{-\frac{d-k}{2}}\beta^{\frac{k}{2}} = \beta^{\frac{k}{2}}\Big[\frac{d-k\beta}{d-k}\Big]^{\frac{d-k}{2}}$$
$$= \beta^{\frac{k}{2}}\Big[1 + \frac{(1-\beta)k}{d-k}\Big]^{\frac{d-k}{2}}$$

Now we show the second inequality of part(II):

$$\beta^{\frac{k}{2}}\Big(1 + \frac{(1-\beta)k}{d-k}\Big)^{\frac{d-k}{2}} = exp\Big[\frac{k}{2}ln(\beta) + \frac{d-k}{2}ln\big(1 + \frac{(1-\beta)k}{d-k}\big)\Big]$$
$$\underset{(63)}{\leq} exp\Big[\frac{k}{2}ln(\beta) + \frac{d-k}{2}\frac{(1-\beta)k}{d-k}\Big]$$
$$= exp\Big[\frac{k(ln(\beta)+1-\beta)}{2}\Big]$$

which finishes part (II). Part (I) is shown in the same way and therefore omitted. □

Proof Johnson-Lindenstrauß Theorem. Let $v_i \in V$ and v_i' be the projection of v_i into a k-dimensional subspace S. To apply lemma 3 we define $L = \|v_i - v_i'\|^2$

and $\mu = \frac{k}{d}\|v_i - v_i'\|^2$ which leads to:

$$\begin{aligned}
&\mathbb{P}(L \leq (1-\epsilon)\mu) \leq exp\Big(\frac{k(1-(1-\epsilon)+ln(1-\epsilon))}{2}\Big) \\
&\underset{\text{Appendix(62)}}{\leq} exp\Big(\frac{k}{2}\Big[\epsilon - \Big(\epsilon + \frac{\epsilon^2}{2}\Big)\Big]\Big) = exp\Big(-\frac{k\epsilon^2}{4}\Big) \\
&\underset{\text{by assumption}}{\leq} exp(-2ln(n)) = \frac{1}{n^2}
\end{aligned}$$

From part (II) of Lemma 3 we get:

$$\mathbb{P}(L \geq (1+\epsilon)\mu) \leq \frac{1}{n^2} \tag{35}$$

Defining $f(v_i) = \sqrt{\frac{d}{k}}v_i'$ and plugging in leads to:

$$\begin{aligned}
&\mathbb{P}\Big(\|f(v_i) - f(v_j)\|^2 \leq (1-\epsilon)\frac{k}{d}\|v_i - v_j\|^2\Big) \\
&= \mathbb{P}\Big(\|v_i' - v_j'\|^2 \leq (1-\epsilon)\|v_i - v_j\|^2\Big) \leq \frac{1}{n^2}
\end{aligned}$$

and

$$\mathbb{P}\Big(\|v_i' - v_j'\|^2 \geq (1+\epsilon)\|v_i - v_j\|^2\Big) \leq \frac{1}{n^2} \tag{36}$$

So the chance that a pair of points lies outside of the interval $[(1-\epsilon),(1+\epsilon)]$ is at most $\frac{2}{n^2}$. Combining all n points gives the probability that for the map

f at least one point is outside of the interval

$$\frac{n(n-1)}{2}\frac{2}{n^2} = 1 - \frac{1}{n} \tag{37}$$

The probability of the complement (that every point is inside of the interval) is therefore at least $\frac{1}{n}$. This shows that the map can be found by repeating the projection n times and therefore in randomized polynomial time. [7] □

In terms of matrices the result from [8] is of particular interest because it gives even higher probabilities for finding a good approximation by random projection and will be used in the stated algorithm.

Theorem 6

Let $A \in \mathbb{R}^{m\times n}$, $0 < \epsilon \leq 1$ and $S \in \mathbb{R}^{r\times n}$ matrix with iid zero-mean and entries $+1$ and -1 with $r = O(\frac{k}{\epsilon} + klogk)$. Furthermore let A_k be the best rank k approximation to A and $A_{S^T,k}$ be the best rank k approximation of the projected matrix AS^T. Then with probability at least $\frac{1}{2}$ the following holds:

$$\|A - A_{S^T,k}\|_F \leq (1+\epsilon)\|A - A_k\|_F \tag{38}$$

Proof. Without proof □

Last in this section we will present an algorithm which combines the ideas of the Nyström method and Johnson-Lindenstrauß Theorem:

Algorithm 1

Input: Covariance Matrix $K_{n\times n}$, dimension of reduction m, dimension of Johnson-Lindenstrauß mapping reduction I

Output: Covariance approximation of the form $K_{n\times m}(K_{m\times m})^{-1}K_{m\times n}$

1. Simulate $\Omega_{n\times I}$ with $iid \sim\ N(0,1)$ and calculate

$$P_{n\times I} = K_{n\times n}\Omega_{n\times I} \tag{39}$$

2. Calculate eigendecomposition of $P_{n\times I}$ and form matrix $\Phi_{n\times m}$ with m eigenvectors of largest eigenvalues.

3. $\bar{K}_{m\times m} = \Phi^T_{m\times n}K_{n\times n}\Phi_{n\times m}$

4. Choleski factorization gives

$$\bar{K}_{m\times m} = B_{m\times m}B^T_{m\times m} \tag{40}$$

5. Set the Nyström factor $C_{n\times m} = K_{n\times n}\Phi^T_{n\times m}(B^T_{m\times m})^{-1}$

6. Singular value decomposition of $C_{n\times m}$ yields

$$C_{n\times m} = U_{n\times m}\Sigma_{m\times m}V^T_{m\times n} \tag{41}$$

Note 8

Now we can approximate $K_{n\times n}$ by $\tilde{K}_{n\times n} = C_{n\times m}C^T_{m\times n}$ which leads to two different representations:

$$\tilde{K}_{n\times n} = C_{n\times m}C^T_{m\times n} = U_{n\times m}\Sigma_{m\times m}V^T_{m\times n}V_{n\times m}\Sigma_{m\times m}U^T_{m\times n}$$
$$= U_{n\times m}\Sigma^2_{m\times m}U^T_{m\times n}$$

and

$$\tilde{K}_{n\times n} = C_{n\times m}C^T_{m\times n} = K_{n\times n}\Phi^T_{n\times m}\big(B^T_{m\times m}\big)^{-1}\big(B_{m\times m}\big)^{-1}\Phi_{m\times n}K_{n\times n}$$
$$= K_{n\times n}\Phi^T_{n\times m}\big(B_{m\times m}B^T_{m\times m}\big)^{-1}\Phi_{m\times n}K_{n\times n}$$
$$= K_{n\times n}\Phi^T_{n\times m}\bar{K}^{-1}_{m\times m}\Phi_{m\times n}K_{n\times n}$$
$$= K_{n\times n}\Phi^T_{n\times m}\big(\Phi^T_{m\times n}K_{n\times n}\Phi_{n\times m}\big)^{-1}\Phi_{m\times n}K_{n\times n}$$

So we have the familiar form where we can by adding a noise variance σ^2 use the Woodbury Matrix Identity.

2.3 Estimation of Hyperparameter

In this section techniques for learning the parameters of the covariance function from the data are developed. From the data one can get different estimators for the parameters by choosing different knots, different starting points for simulation or just because of the nature of random variables we need to simulate in the procedure. By generating enough of the parameters we can derive all the characteristics of a distribution, even though we do not know the density function. For example with the law of large numbers we can estimate the mean $\hat{\mu} = \mathbb{E}(x) = \frac{1}{n}\sum_{i=1}^{n} x_i$ etc.

2.3.1 Maximum likelihood function

One can obtain the parameters just by maximizing the marginal likelihood function. For different choices of data included to "train" the parameters we obtain different parameter estimators from which we can derive the distribution of the parameters.

What needs to be maximized for prediction purposes is the probability of the

observed values $y_1, ..., y_n$, given $t_1, ..., t_n$ and a certain structure of the covariance kernel. We then maximize w.r.t the hyperparameters of the covariance kernel. In section 2.1 we derived that

$$y \sim N(0, \underbrace{K^+_{n\times m}(K^*_{m\times m})^{-1}K^+_{m\times n}}_{=:K_{n\times n}} + \sigma^2 I_{n\times n}) \tag{42}$$

Then we obtain the marginal likelihood:

$$p(y|x) = \frac{1}{\sqrt{(2\pi)^k |K_{n\times n} + \sigma^2 I_{n\times n}|}} exp\big(-\frac{1}{2}(y)^T(K_{n\times n} + \sigma^2 I_{n\times n})^{-1}(y)\big)$$

and the log marginal likelihood

$$logp(y|x) = -\frac{1}{2}(y)^T(K_{n\times n} + \sigma^2 I_{n\times n})^{-1}(y) - \frac{1}{2}log|K_{n\times n} + \sigma^2 I_{n\times n}| - \frac{n}{2}log2\pi$$

Now the log marginal likelihood can be maximized over the hyperparameters of its covariance function.

2.3.2 Sampling

Here we will get familiar with three important techniques to simulate random variables even though we do not fully know their density function and how

to simulate them. First, the acceptance and rejection method enables us to simulate random variables where the density function is known just by simulating two other random variables. Next, with the help of the Gibbs sampling, which uses basic properties of Markov chains, we can simulate random variables when only their conditional distribution is known. Last the Metropolis Hastings Algorithm tells us how to simulate when we know the density function up to a multiplicative factor. By combining these techniques one needs to assume a prior distribution of the hyperparameters and then by multiple iterations of sampling and updating of the distribution parameters one gets closer to the true distribution of the hyperparameters.

Acceptance and Rejection Method

The acceptance and rejection method enables us to simulate random variables from any density function without any transformations of the density function (in contrast to the inversion method where the inverse of the cumulative distribution function is needed and can be tedious to find in some cases). The basic idea of this method is to sample from another density function g where it is easy to sample from and whose support includes the support of f. One can then accept a sample from g under some condition as

a sample from f. If we look at the quotient $\frac{f(x)}{g(x)}$ we see that for $\frac{f(x)}{g(x)} > 1$ it is more likely that x is generated from f than from g. If we further "normalize" this quotient by $c = sup_x \frac{f(x)}{g(x)}$ we have that $0 \leq \frac{f(x)}{cg(x)} \leq 1$. Now a sample y, generated from g, will be accepted as sample from f if $\frac{f(y)}{cg(y)} \geq U$ with $y \sim g$ and $U \sim U(0,1)$.

Algorithm 2

Input: function $f(x)$ from which we want to sample, function $g(x)$ whose support includes the support of $f(x)$ and we are able to sample from, constant c with $sup_x[\frac{f(x)}{g(x)}] \leq c$

Output: $X \sim f(x)$

1. Generate $Y \sim g$
2. Generate $U \sim U(0,1)$
3. If $(\frac{f(Y)}{cg(Y)} \geq U)$ then return $X = Y$

   else go to step 1

We have to show two things about the algorithm: 1. The output of this algorithm is sampled from f and 2. the algorithm terminates with probability 1.

First some notation:

Let N be the stopping time of the algorithm and R_n the event, that the algorithm terminates after iteration n respectively $y \sim g$ is rejected $n-1$ times as a sample from f. Formally:

$$R_n = \Big\{\omega \in \Omega : U_1(\omega) > \frac{f(Y_1(\omega))}{cg(Y_1(\omega))}, ..., U_{n-1}(\omega) > \frac{f(Y_{n-1}(\omega))}{cg(Y_{n-1}(\omega))}, U_n(\omega) \leq \frac{f(Y_n(\omega))}{cg(Y_n(\omega))}\Big\} \tag{43}$$

If we define p as the probability that the algorithm ends in one certain iteration, we have:

$$p = \mathbb{P}\Big(\Big\{\omega \in \Omega : U(\omega) \leq \frac{f(Y(\omega))}{cg(Y(\omega))}\Big\}\Big) = \frac{1}{c} \tag{44}$$

If we look at R_n as a random variable, we see that R_n has a geometric distribution with 'winning' probability p and hence $\mathbb{P}(R_n) = p(1-p)^{n-1}$.

Lemma 4

Algorithm 2 terminates.

Proof.

$$\mathbb{P}(N < \infty) = \mathbb{P}(R_1 \cup R_2 \cup R_3 \cup ...) \underset{R_i \cap R_j = \emptyset \text{ for } i \neq j}{=} \sum_{n=1}^{\infty} \mathbb{P}(R_n) = \sum_{n=1}^{\infty} p(1-p)^{n-1}$$

$$= \sum_{n=0}^{\infty} p(1-p)^n \underset{\text{geometric sum}}{=} \frac{p}{1-(1-p)} = 1$$

□

Lemma 5

Output of algorithm 2 is sampled from f.

Proof.

$$\mathbb{P}(X \leq x) = \sum_{n=1}^{\infty} \mathbb{P}(X \leq x \cap R_n)$$

$$= \sum_{n=1}^{\infty} \mathbb{P}\Big(Y_n \leq x, U_1(\omega) > \frac{f(Y_1(\omega))}{cg(Y_1(\omega))}, ..., U_{n-1}(\omega) > \frac{f(Y_{n-1}(\omega))}{cg(Y_{n-1}(\omega))}, U_n(\omega) \leq \frac{Y_n(\omega)}{cg(Y_n(\omega))}\Big)$$

$$= \sum_{n=1}^{\infty} \mathbb{P}\Big(Y_n \leq x, U_n(\omega) \leq \frac{Y_n(\omega)}{cg(Y_n(\omega))}\Big)\mathbb{P}\Big(U_1(\omega) > \frac{f(Y_1(\omega))}{cg(Y_1(\omega))}, ..., U_{n-1}(\omega) > \frac{f(Y_{n-1}(\omega))}{cg(Y_{n-1}(\omega))}\Big)$$

$$= \sum_{n=1}^{\infty} \mathbb{P}\Big(Y_n \leq x, U_n(\omega) \leq \frac{Y_n(\omega)}{cg(Y_n(\omega))}\Big)(1-p)^{n-1}$$

$$= \sum_{n=1}^{\infty} \Big[\int_{-\infty}^{x} \mathbb{P}\big(U_n \leq \frac{f(y)}{cg(y)}\big) g(y) dy\Big](1-p)^{n-1}$$

$$= \sum_{n=1}^{\infty} \Big[\int_{-\infty}^{x} \frac{f(y)}{cg(y)} g(y) dy\Big](1-p)^{n-1} = \sum_{n=1}^{\infty} \frac{F(x)}{c}(1-p)^{n-1}$$

$$= \frac{F(x)}{c} \sum_{n=1}^{\infty} (1-p)^{n-1} \underset{\text{geometric sum}}{=} \frac{F(x)}{cp} \underset{(44)}{=} F(x)$$

□

Gibbs sampling

In Gibbs sampling we basically draw samples from the conditional distributions of the random variables and rely that they will (after a "few" iterations) converge to their joint distribution. To give the reasoning for that we need to explore some basic concepts of Markov Chains. Markov Chains are stochastic processes where the present state of the chain carries all the information we need to predict the future, formally:

Definition 2

Let $(\Omega, \mathcal{B})$ be a probability space and $X_n : \Omega \mapsto \mathcal{S}$ for $n = 0, 1, 2, ...$ Then X_n is called a Markov Chain if

$$\mathbb{P}(X_{n+1} \in A | X_n = x_n, ..., X_1 = x_1) = \mathbb{P}(X_{n+1} \in \mathcal{A} | X_n = x_n) \text{ a.s.}$$

for all $n > 0$ and $A \in \mathcal{B}(S)$. We call $\mathbb{P}(X_{n+1} \in \mathcal{A} | X_n = x_n)$ the transition probability as it gives us the probability to go from $X_n = x_n$ to $X_{n+1} \in \mathcal{A}_{n+1}$.

For $n = 0$ we have $\mathbb{P}(X_0 \in \mathcal{A}) = \pi(\mathcal{A})$ which we also called the initial distribution. A Markov Chain is called stationary, if the joint distribution of $\{X_n, ..., X_{n+k}\}$ is independent of n and k. Therefore $\mathbb{P}(X_{n+1} \in \mathcal{A}|X_n = x)$ is independent of n and we can write $\mathbb{P}(A|x)$.

Note 9

We make use of regular conditional distributions as $\mathbb{P}(X = x) = 0$ in a continuous state space but we nevertheless like to make use of conditional distributions $\mathbb{P}(X_{n+1} \in \mathcal{A}|X_n = x_n)$.

Note that a Markov Chain is fully specified by its transition probability and initial distribution. Our goal is to construct a Markov Chain which converges to our desired multivariate distribution. The questions is now, how do we construct such a Markov Chain? To answer this question we define the following:

Definition 3 (Equilibrium distribution)

Let $\mathbb{P}_x(X_n \in \mathcal{A}) = \mathbb{P}(X_n \in \mathcal{A}|X_0 = x)$ be the n steps transition probability starting at x. $\lim_{n\to\infty} \mathbb{P}_x(X_n \in \mathcal{A}) = \bar{\pi}(\mathcal{A})$ is called the equilibrium distribution, if it exists.

Definition 4 (Invariant distribution)

We call π an invariant distribution if it satisfies

$$\pi(\mathcal{A}) = \int \mathbb{P}_x(X_n \in \mathcal{A}) d\pi(x) \tag{45}$$

Lemma 6

Let $\{X_n\}_{n=0}^{\infty}$ be a stationary Markov Chain. The invariant distribution π and equilibrium distribution $\bar{\pi}$, if it exists, are the same.

Proof. Suppose the limiting distribution $\pi()$ on $\mathcal{B}(S)$ exists such that

$$\lim_{n\to\infty} \mathbb{P}_x(X_n \in \mathcal{A}) = \bar{\pi}(\mathcal{A}) \tag{46}$$

for all $x \in \mathcal{S}$ and $A \in \mathcal{B}(S)$.

Then

$$\begin{aligned}
&\mathbb{P}_\pi(X_n \in A) = \int \mathbb{P}(X_n \in A | X_0 = x) d\pi(x) = \int \mathbb{P}_x(X_n \in A) d\pi(x) \\
&\implies \lim_{n\to\infty} \mathbb{P}_\pi(X_n \in \mathbb{A}) = \int \lim_{n\to\infty} \mathbb{P}_x(X_n \in A) d\pi(x) \\
&= \bar{\pi}(A) \int d\pi(x) = \bar{\pi}(A)
\end{aligned}$$

for all π.

Now, looking at the next state of the Markov chain we get:

$$\mathbb{P}_\pi(X_{n+1} \in A) = \int \mathbb{P}(X_{n+1} \in A | X_n = x) d\mathbb{P}_\pi(X_n = x)$$
$$= \int p(A|x) d\mathbb{P}_\pi(X_n = x)$$

and

$$\underbrace{\lim_{n\to\infty} \mathbb{P}_\pi(X_{n+1} \in \mathcal{A})}_{=\bar{\pi}(A)} = \int p(A|x) d \lim_{n\to\infty} \mathbb{P}_\pi(X_n = x)$$
$$= \int p(A|x) d\bar{\pi}(x)$$

□

Note 10

The motivation for the preceding lemma was to shift the problem of finding a distribution to which the Markov Chain converges to finding an invariant distribution.

Now suppose that $X \sim \pi$. We condition X on its own values, namely $Y = f(X)$ for some $f : \mathcal{S} \mapsto \mathcal{R}^k$ for some k.

$$\implies p(A|x) = \mathbb{P}(X \in A | Y = y) \tag{47}$$

Lemma 7

$p(A|x)$ is a transition probability with invariant distribution π, i.e.

$$\int p(A|x)d\pi(x) = \pi(A) \tag{48}$$

Proof. Let $B(x) = \{u : f(u) = f(x)\}$ be the set of elements $u \in \mathcal{S}$ which are mapped to the same element $\in \mathcal{R}^k$ as x and $I_A(x)$ be the indicator function:

$$I_A(x) = \begin{cases} 1 & \text{if } x \in A \\ \\ 0 & \text{else} \end{cases} \tag{49}$$

Then we have:

$$\begin{aligned}
&\int p(A|x)d\pi(x) \underset{X\sim\pi \text{ and Bayes Theorem}}{=} \int \Big(\frac{\pi(A \cap B(x))}{\pi(B(x))}\Big)\pi(x)dx \\
&= \int \Big(\int I_A(u) I_{B(x)}(u)\pi(u)du\Big)\frac{\pi(x)}{\pi(B(x))}dx \\
&\underset{\text{Fubini}}{=} \int I_A(u)\Big(\int \frac{I_{B(x)}(u)\pi(x)}{\pi(B(x))}dx\Big)\pi(u)du \\
&\underset{u\in B(x) \iff x\in B(u)}{=} \int I_A(u) \underbrace{\Big(\int \frac{I_{B(u)}(x)\pi(x)}{\pi(B(u))}dx\Big)}_{=1} \pi(u)du \\
&= \int I_A(u)\pi(u)du = \pi(A)
\end{aligned}$$

□

Finally, for $x = (x_1, ..., x_k) \in \mathcal{S} \subset \mathcal{R}^k$ we define $f_i(x) = (x_1, ..., x_{i-1}, x_{i+1}, ..., x_k)$ and obtain the algorithm for Gibbs sampling:

Algorithm 3

Input: starting values $(x_1^0, x_2^0, ..., x_n^0)$ for every variable, conditional distributions $\pi(x_j|x_1, ..., x_{j-1}, x_{j+1}, ..., x_k)$ for all $j = 1, .., k$, integer $m =$ number of iterations

Output: samples from $\pi(x_1, ..., x_k)$.

for $i = 1$ to m

generate $x_1^i \sim \pi(x_1|x_2^{i-1}, x_3^{i-1}, ..., x_k^{i-1})$

generate $x_2^i \sim \pi(x_2|x_1^i, x_3^{i-1}, ..., x_k^{i-1})$

$\vdots$

generate $x_k^i \sim \pi(x_k|x_1^i, x_2^i, ..., x_{k-1}^i)$

end for

return $(x_1^m, ..., x_k^m)$

Note 11

One can use the Gibbs sampling in two different ways: Either one starts with some x and accepts x^m for some m as a random variable sampled from π and starts all over again to sample the next one. Another approach is to sample

x^m for some large m and accept all x from a certain point, say x^j with $j < m$. The simulated values until j are also called burn-in period. The choice of the length of the burn-in period is not obvious and generally longer ones are preferable. For discussion see [9].

Metropolis Hastings Algorithm

It is often the case that a density function is given up to a multiplicative factor, i.e.

$$\pi(x) \propto f(x) \tag{50}$$

where $\pi(x)$ is the target density we want to sample from and $f(x)$ is an unknown density function. Finding the normalizing constant which ensures that $c \int \pi(x)dx = 1$ is not always easy. The Metropolis Hastings Algorithm gives us a solution for this problem by enabling sampling from density functions which do not integrate to one. Metropolis Hastings algorithm is a Monte Carlo Markov Chain (MCMC) algorithm. So it relies on many iterations and also on the properties of Markov Chains. The basic idea is that the Markov Chain moves through the domain of the density function and stays more often in regions with high probability and less often in regions of the domain with low probability. The states of the Markov Chain then build

the samples. The decision if the Markov Chain moves to another state is similar to the acceptance and rejection method, where a certain sample was accepted by comparing it to a quotient of density functions. Although the density is known up to a multiplicative factor, the quotient $\frac{\pi(y)}{\pi(x)}$ equals the quotient of the "real" density. If the quotient is > 1 the generated sample is more probable than the current one and is accepted as the next state of the Markov Chain. If it is smaller than 1, then it is compared with a random variable $U \sim U(0,1)$ and accepted in case $\frac{\pi(y)}{\pi(x)} > U$. It is rejected otherwise. The question left is from which density the samples are generated. We call this density $p(\cdot|x)$ also candidate generating conditional density and will derive it in the following:

There are 4 requirements for $p(\cdot|x)$:

1. $\int p(y|x) = 1$ for π almost all x

2. $p(\cdot|x)$ converges to π, or equivalent:

$$\int p(A|x)\pi(x)dx = \pi(A) \text{ for all } A \in \mathcal{B}(S) \tag{51}$$

3. The Markov chain with transition probability $p(y|x)$ and initial probability $\pi(x)$ has to be ergodic, i.e. it is possible to go from every state

to every other state.

4. It is easy to sample from $p(\cdot|x)$.

The strategy to construct such a $p(y|x)$ is to start with some $q(y|x)$ and transform it in two steps, so that it satisfies the mentioned conditions.

First: It is possible that

$$\int q(y|x)dy < 1 \tag{52}$$

which means that at current state x it may not move to another state y and is stuck. Therefore it is added another term $r(x)$ and we get:

$$p(dy|x) = q(y|x)dy + r(x)\delta_x(dy) \tag{53}$$

where $\delta_x(dy) = 1$ if $x \in dy$ and 0 otherwise and $r(x) = 1 - \int q(y|x)dy$. Hence we can interpret $r(x)$ also as the probability that the chain stays at x.

Second: It is possible that

$$\pi(x)p(y|x) \neq \pi(y)p(x|y) \tag{54}$$

which is needed for the second condition. w.l.o.g. let

$$\pi(x)p(y|x) > \pi(y)p(x|y) \tag{55}$$

On the left hand side we have the probability of being at x and moving to y and on the left hand side vice versa. We remedy this problem by multiplying with an extra probability $\alpha(x|y)$ and $\alpha(y|x)$, also called 'probability of move'. On the right hand side we make $\alpha(x|y)$ as large as possible, namely 1, to get:

$$\begin{aligned} &\pi(x)q(y|x)\alpha(y|x) = \pi(y)q(y|x)\alpha(x|y) \underset{\alpha(x|y)=1}{=} \pi(y)q(x|y) \\ &\implies \alpha(y|x) = \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)} \end{aligned}$$

Therefore the rule for choosing $\alpha(y|x)$ is:

$$\alpha(y|x) = \begin{cases} min\Big(\frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}, 1\Big) & \text{if } \pi(x)q(y|x) > 0 \\ 1 \quad \text{otherwise} \end{cases} \tag{56}$$

Finally the transition kernel $p_{MH}(dy|x)$ is given by:

$$p_{MH}(dy|x) = q(y|x)\alpha(y|x)dy + \Big[1 - \int_{\mathcal{S}} q(y|x)\alpha(y|x)dy\Big]\delta_x(dy) \tag{57}$$

Before the algorithm is stated let's be more specific about the choice of $q(y|x)$. Here are three options:

1. Random Walk Chain

   Set $q(y|x) = q_1(y - x)$ with $q_1(\cdot)$ a multivariate density, for example multivariate normal. The generated sample will then be added to the current state, i.e. let x be the current state and $Z \sim q_1$. The new state y is obtained by $y = x + z$

2. Independence chain

   The next state y is independent of the current state x, i.e. $q(y|x) = q_2(y)$.

3. Independent choice derived from $\pi(x)$

   If $\pi(x) \propto \Psi(x)h(x)$ with $h(x)$ a known density and $\Psi(x)$ uniformly bounded, set $q(y|x) = h(y)$ and $a(y|x) = min\left\{\frac{\Psi(y)}{\Psi(x)}, 1\right\}$.

Last, for stating the algorithm we need a little bit notation: If $r(x) \neq 0$ then $Q(y|x)$, the cdf of $q(y|x)$ is not continuous respectively has jumps. Let $y_1, ..., y_k$ be the jump points and $\gamma(y_1|x), ...\gamma(y_k|x)$ the size of the jumps. Last, $\gamma(x) = \sum_{i=1}^{k} \gamma(y_i|x)$

Algorithm 4 (Metropolis Hastings Algorithm)

Input: transition kernel $p_{MH}(dy|x)$, starting point x_0, n = number of samples needed, jump points $(y_1, ..., y_k)$ and their jump sizes $\gamma(y_1|x), ...\gamma(y_k|x)$

Output: states of the Markov Chain respectively samples $\sim \pi(x)$

if $r(x) = 0$

  while $(i < n)$

    generate $U \sim U(0,1)$

    generate $y \sim p_{MH}(dy|x)$

    if $(U \leq \alpha(y|x))$

      set $x_{i+1} = y$ and $i = i + 1$

  return $(x_1, ..., x_n)$

else

  while (i<n)

    generate $U \sim U(0,1)$

    if $(U < \gamma(x))$

      select one of $\{y_1, ..., y_k\}$ with probability proportional

      to their length $\gamma(y_1|x), ...\gamma(y_k|x)$ and set $x_{i+1} = y_j$ and $i = i + 1$

    else

    if $(U \leq \alpha(y|x))$

    set $x_{i+1} = y$ and $i = i + 1$

return $(x_1, ..., x_n)$

3 Application

As an example application the random projection method (2.2.1) and marginal likelihood maximization (2.3.1) were implemented in R and tested with the data set SARCOS. The code is given in Appendix B. Sarcos is a robot arm with 7 joints. For every joint position, velocity and acceleration is measured and served as the independent variables to predict the 7 torques. The connection is not a static one because of the hydraulic nature of the arm. This made it appropriate for statistic modeling. Furthermore the size of the data set suggested use of approximation methods: 44484 x 28 observations. Furthermore a test set of 4449 x 28 observations is used for cross validation. Here the 21 inputs were used to predict the first torque. The data set was already centered, i.e. the mean equals 0 in every dimension. This let us also choose 0 as a mean function. Furthermore we made use of the squared exponential function, which we restate here:

$$cov(x, y) = \frac{1}{\theta_1} * exp\Big[\frac{(x-y)^2}{\theta_2}\Big] \tag{58}$$

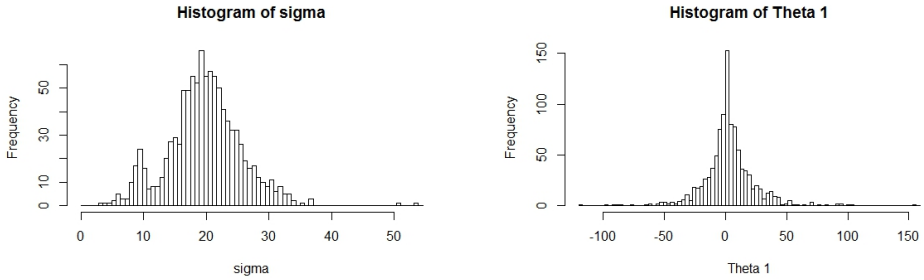
In the results of the computation the starting point for the hyperparameter estimation θ_1, θ_2 and σ^2 were crucial. The conjugate gradient method was used for maximization of the log marginal likelihood. Prior examination of the data let small values for the hyperparameters be reasonable, roughly $\theta_1 \approx \frac{1}{9000}$, $\theta_2 \approx \frac{1}{10000}$ and $\sigma \approx 1$. So as a starting point 1 was chosen for all parameters. Out of the 44484 observation $n = 1000$ were used together with $m = 400$ knots. The estimations were made 1000 times and the plots together with some important statistic measures are shown below:

	mean	median	95% confidence interval	95% HPD
σ	8.984	17.3	[8.366 , 31.562]	[8.270 , 30.385]
θ_1	2.730	2.000	[-37.75 , 43.607]	[-37.729 , 43.700]
θ_2	14.610	9.274	[0.662 , 57.923]	[0.143 , 46.774]

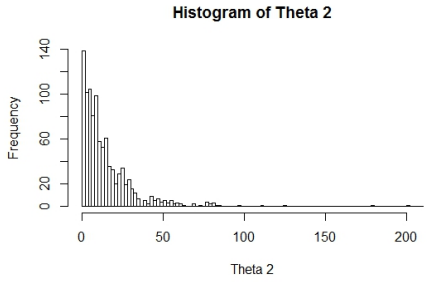
Although the results seem reasonable, predicting with those parameters (for example the mean of every estimator) gives poor results. Estimators were expected to be tighter concentrated and much smaller. For computational reasons a slight modification of the covariance function was made:

$$cov(x, y) = \theta_1 * exp\big((x - y)^2\theta_2\big) \tag{59}$$

where the hyperparameters were inverted. Starting values were now 9000



(a) σ^2 (b) θ_1

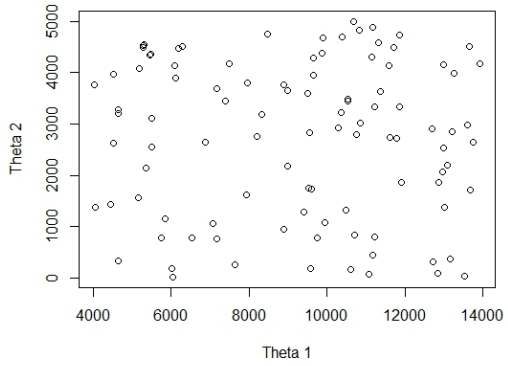


(c) θ_2

Figure 1: Histogram of $\sigma^2, \theta_1, \theta_2$ (1)

for θ_1 and 10000 for θ_2. The results indicated that the conjugate gradient method gets stuck in a local extrema. So the next idea was to use a random grid of starting values as shown below:

From every point the log marginal likelihood was maximized and the 100 results were compared and the largest one was kept. The obtained distribution of the parameters is shown below:

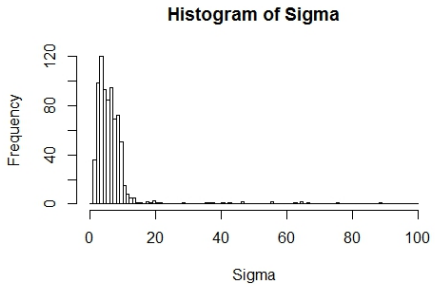


	mean	median	95% confidence interval	95% HPD
σ	717.587	6.489	[1.850 , 5463.814]	[1.263, 2455.291]
θ_1	7612.226	8272.148	[-2266.15 , 14039.17]	[-1301.175 , 14574.128]
θ_2	2083.991	2081.892	[-1710.406 , 7990.827]	[-1301.175 , 14574.128]

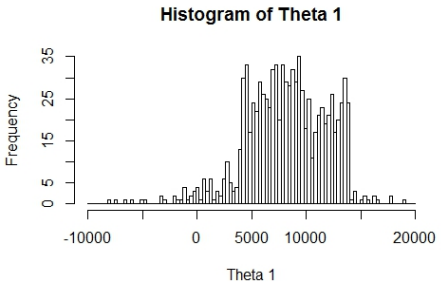
With the median of every parameter predictions were made (shown in figure 3 and 4). The black points are the actual values and the red ones the predicted. Those observations were not included in the training of the parameters. x_1 and x_{17} correspond to the 1^{st} respectively 17^{th} column of the data and y to the 22^{nd}. The standardized mean squared error is fund to be 0.0265.

Note 12

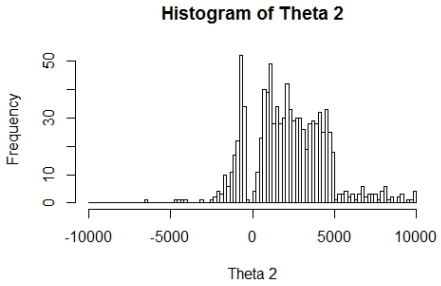
One can observe that the mean is no robust estimator as between the mean and median of σ^2 there is a big difference. This is the reason predictions were made with the median. Also observe that those computations are not computationally efficient anymore. The conjugate gradient method got stuck in



(a) σ^2



(b) θ_1



(c) θ_2

Figure 2: Histogram of $\sigma^2, \theta_1, \theta_2$ (2)

local extrema and did not find the global ones. We assume that because extrema were often fund close to starting points. Other maximization methods for example Broyden-Fletcher-Goldfarb-Shanno algorithm showed no superiority. Nevertheless with the help of the grid we were able to make use of the log marginal likelihood maximization and predictions gave reasonable results.

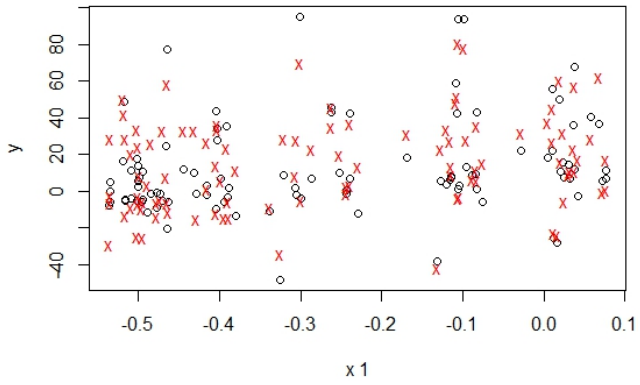


Figure 3: x_1 vs y and predictions

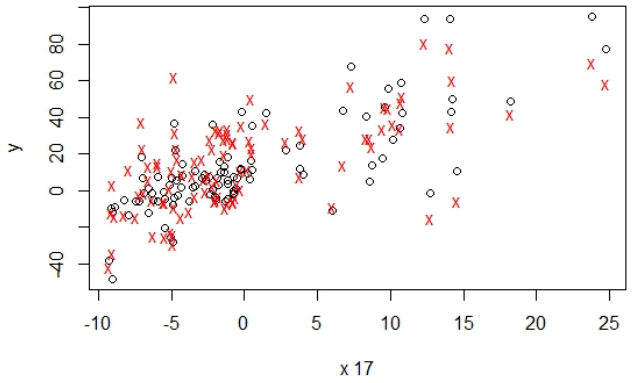


Figure 4: x_{17} vs y and predictions

4 Conclusion and outlook

Gaussian Processes are a powerful tool for predictions. What is particular interesting and surprising is that random projection methods perform very well

[5]. They are computationally undemanding and are also easy to interpret which in consequence makes them quite useful.

Conclusion (Limitations of Gaussian Process Regression)

Although they perform well in predictions, Gaussian Process have no easy to interpret statistic which tells us about the relationship of the variables or graphical output which enables a deeper understanding of the estimated relationship in high dimensions.

Conclusion (Outlook)

Besides other machine learning algorithms one can also use Gaussian Processes for classification problems. Furthermore we did not explore any computational issues. Recent development makes use of parallel computing. Thereby one needs to split the necessary computations in parts and let it be computed by different processors which in consequence speeds up the computations. Last, there are more parts of Gaussian Process Regression which can be speeded up with the help of approximation methods, for example singular value decomposition, matrix multiplications etc. [8].

A Mathematical Background

Definition 5

The Frobenius norm of a $n \times m$ matrix A is given by:

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} |a_{i,j}|^2} \tag{60}$$

Lemma 8

Let x and y be vectors with multivariate normal distribution and Matrix A,B,C of appropriate dimension:

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim N\left(\vec{0}, \begin{bmatrix} A & C \\ C^T & B \end{bmatrix}\right)$$

Then the conditional distribution of x given y is:

$$x|y \sim N(CB^{-1}y, A - CB^{-1}C^T) \tag{61}$$

Power series of logarithm gives:

$$ln(1+x) = \sum_{k=1}^{\infty}(-1)^{k+1}\frac{x^k}{k} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + ... \underset{x<0}{\leq} x - \frac{x^2}{2} \qquad (62)$$

Furthermore for the logarithm it holds that

$$ln(x) \leq x - 1 \qquad (63)$$

B R Code

```
#read in the data from local PC
data=read.table("C:/.../Sarcos_inv.csv", header=FALSE,sep=",")
test_data=read.table("C:/.../Sarcos_inv_test.csv",
header=FALSE,sep=",")


#number of observations to incorporate
n=1000
#number of observations for covariance matrix random projection
m=400
```

```
#extract n random samples
randData=sample(1:44484,n,replace=FALSE)
X_n=as.matrix(data[randData,1:21])
Y_n=data[randData,22]

#extract 100 observations from the test data
X_p=as.matrix(test_data[1:100,1:21])
Y_p=test_data[1:100,22]
I=diag(1,n,n)

#covariance function without noise variance sigma
covfunc1=function(hyper,x1,x2){
  K=matrix(0,dim(x1)[1],dim(x2)[1])
  for(i in 1:dim(x1)[1]){
    for(j in 1:dim(x2)[1]){
      K[i,j]=(hyper[2]^(1))*exp(-sum((x1[i,]-x2[j,])^2)*
                                (hyper[3]^(-1)))
    }
  }
```

```
  return(K)
}

#covariance function with noise variance sigma
covfunc2=function(hyper,x1){
  K=matrix(0,dim(x1)[1],dim(x1)[1])
  for(i in 1:dim(x1)[1]){
    for(j in 1:dim(x1)[1]){
      K[i,j]=(hyper[2]^(1))*exp(-sum((x1[i,]-x1[j,])^2)*
                                (hyper[3]^(-1)))+(hyper[1]^2)*(i==j)
    }
  }
  return(K)
}

#Marginal Likelihood of Y using Woodbury Identity
marginal_likelihood=function(hyper){

  random_vector=sample(1:n,m,replace=FALSE)
```

```
  permutation_matrix=I[random_vector,]

#Those observations are building the knots
  X_m=permutation_matrix %*% X_n

#rest of the observations
  X_n_minus_m=I[-random_vector,] %*% X_n
  Y_n_minus_m=I[-random_vector,] %*% Y_n

  K_star_mxm=covfunc2(hyper,X_m)
  K_star_mxm_inv=solve(K_star_mxm)
  K_plus_n_minus_mxm=covfunc1(hyper,X_n_minus_m,X_m)

  cov=K_plus_n_minus_mxm %*% K_star_mxm_inv %*%
        t(K_plus_n_minus_mxm)+diag(hyper[1]^2,n-m,n-m)

  #Woodbury Matrix Identity
  cov_inv=hyper[1]^(-2)*(diag(1,(n-m),(n-m))-
        (K_plus_n_minus_mxm %*% solve((hyper[1]^2)*
```

```
        K_star_mxm+(t(K_plus_n_minus_mxm) %*%
        K_plus_n_minus_mxm)) %*% t(K_plus_n_minus_mxm)))

  # return log marginal likelihood
  return((-0.5*t(Y_n_minus_m) %*% cov_inv %*% (Y_n_minus_m))
        -0.5*determinant(cov,logarithm=TRUE)[[1]][[1]]
        -((n-m)/2)*log(2*pi))
}

# this function gives back k estimators of the
# hyperparameters and uses the marginal_likelihood function
hyperparameter_estimation=function(k){

  output=matrix(1,k,3)

  for(l in 1:k){     #in every loop one estimator is made
    for(b in 1:100){        #100 estimators are made and compared
      i=runif(1,4000,14000) #random grit
      j=runif(1,1,5000)
```

```
        o=runif(1,1,10)

        try({temp=optim(c(o,i,j),marginal_likelihood,
                    control=list(fnscale=-1),method="CG")[[1]]})

        if(marginal_likelihood(temp)>
                              marginal_likelihood(output[l,])){
          output[l,]=temp
        }
        else{}
      }
    }

    return(output)

}

hyperparameter=hyperparameter_estimation(1000)
```

```
#saving the simulated values
write.table(hyperparameter, "C:/.../hyperparameter.csv",
sep=",", append=TRUE,row.names = FALSE, col.names = FALSE)


# predicting:
random_vector=sample(1:n,m,replace=FALSE)
permutation_matrix=I[random_vector,]
#Those observations are building the knots
X_m=permutation_matrix %*% X_n


K_plus_pxn=covfunc1(hyperparameter,X_p,X_n)
K_plus_pxm=covfunc1(hyperparameter,X_p,X_m)
K_plus_nxm=covfunc1(hyperparameter,X_n,X_m)
K_star_mxm=covfunc2(hyperparameter,X_m)
K_star_mxm_inv=solve(K_star_mxm)
K_tilde=K_plus_pxm %*% K_star_mxm_inv %*% t(K_plus_pxm)
cov=K_tilde+diag(1,dim(X_p)[1],dim(X_p)[1]) %*%
        (covfunc2(hyperparameter,X_p)-K_tilde)
#Woodbury Identity:
```

```
cov_inv=(hyperparameter[1]^(-2))*(diag(1,n,n)-K_plus_nxm %*%
        solve((hyperparameter[1]^2)*K_star_mxm+t(K_plus_nxm) %*%
        K_plus_nxm) %*% t(K_plus_nxm))
predictions=K_plus_pxn %*% cov_inv %*% Y_n


#Plot
plot(X_p[,1],Y_p)
points(X_p[,1],predictions,col="red", pch=4)


# evaluating
SMSE=function(x,y,var){
   return(sum(((x-y)^2)/var)/length(x))
}


SMSE(predictions,Y_p,diag(cov))
```

References

[1] K. G. Coffman and A. M. Odlyzko. *Handbook of Massive Data Sets*, chapter Internet Growth: Is There a "Moore's Law" for Data Traffic?, pages 47–93. Springer US, Boston, MA, 2002.

[2] Gordon Moore. Moore's law (1965). *URL: http://www. intel. com/research/silicon/mooreslaw. htm*, 2014.

[3] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[4] Andrew O Finley, Huiyan Sang, Sudipto Banerjee, and Alan E Gelfand. Improving the performance of predictive process modeling for large datasets. *Computational statistics & data analysis*, 53(8):2873–2884, 2009.

[5] Anjishnu Banerjee. *Scalable Nonparametric Bayes Learning*. PhD thesis, Duke University, 2013.

[6] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Proceedings of the 14th Annual Confer-*

ence on Neural Information Processing Systems, number EPFL-CONF-161322, pages 682–688, 2001.

[7] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[8] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 143–152. IEEE, 2006.

[9] George Casella and Edward I George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.