Gaussian Processes: Questions and Answers

Mike McCourt

Meshfree Seminar
Illinois Institute of Technology
June 4, 2013
Outline

1. Fundamental Problem: Approximation Theory
2. Fundamental Problem: Gaussian Processes
3. Comparing Gaussian Processes and Approximation Theory
4. Hilbert-Schmidt SVD for Gaussian Processes
5. HS SVD Application: Covariance Matrix Determinant
6. HS SVD Application: LOOCV Shape Parameter Optimization
7. Summary
Kernel-based Interpolation

Given data \((x_i, y_i)_{i=1}^N\), use a data-dependent linear function space

\[ s_f(x) = \sum_{j=1}^N c_j K(x, x_j), \quad x \in \Omega \subseteq \mathbb{R}^d \]

with \(K : \Omega \times \Omega \rightarrow \mathbb{R}\) a positive definite reproducing kernel.
Kernel-based Interpolation

Given data \((x_i, y_i)_{i=1}^{N}\), use a data-dependent linear function space

\[
s_f(x) = \sum_{j=1}^{N} c_j K(x, x_j), \quad x \in \Omega \subseteq \mathbb{R}^d
\]

with \(K: \Omega \times \Omega \to \mathbb{R}\) a positive definite reproducing kernel.

To find \(c_j\) solve the interpolation equations

\[
s_f(x_i) = f(x_i) = y_i, \quad i = 1, \ldots, N.
\]

Leads to linear system \(Kc = y\) with symmetric positive definite – often ill-conditioned – system matrix

\[
K_{ij} = K(x_i, x_j), \quad i, j = 1, \ldots, N.
\]
Kernel-based Interpolation

The goal of kernel-based interpolation is to produce this function \( s_f \).

Given the data locations \( \{x_i\}_{i=1}^N \), the function \( s_f \) is defined by the \( c_j \) coefficients.

These coefficients are our *solution* to the interpolation problem.

With them, we can evaluate our interpolant at any location \( x \in \Omega \).

We can also evaluate a linear operator on \( \mathcal{L} \) to \( s_f \) by applying it to the kernels: \( \mathcal{L}K(\cdot, x_i) \).
Gaussian Processes

Definition

$y_x$ is a Gaussian process if, for a choice of distinct locations $\{x_i\}_{i=1}^N$ in $\Omega$, the random vector

\[
y = \begin{pmatrix}
y_{x_1} \\
\vdots \\
y_{x_N}
\end{pmatrix}
\]

has a multivariate normal distribution with mean vector $\mu = E(y)$ and covariance matrix $K = \text{cov}(y, y)$.

We will write $y \sim N(\mu, K)$ to indicate the multivariate normal with mean $\mu$ and covariance $K$. 
Gaussian Processes

Density Function

If \( y \sim N(\mu, K) \), the probability density function is

\[
p(y; \mu, K) = \frac{1}{(2\pi)^{d/2} \det K} \exp \left( -\frac{1}{2} (y - \mu)^T K^{-1} (y - \mu) \right)
\]

This can also be called the likelihood function?
Fundamental Problem: Gaussian Processes

Gaussian Processes

Density Function

If $y \sim N(\mu, K)$, the probability density function is

$$p(y; \mu, K) = \frac{1}{(2\pi)^{d/2} \det K} \exp \left( -\frac{1}{2} (y - \mu)^T K^{-1} (y - \mu) \right)$$

This can also be called the likelihood function?

Therefore, a Gaussian process is uniquely defined by its mean and covariance functions

$$\mu(x) = \mathbb{E} y_x, \quad \text{and} \quad K(x, z) = \text{cov}(y_x, y_z).$$

We will write $y_x \sim GP(\mu, K)$ for such a Gaussian process.
Fundamental Problem: Gaussian Processes

Gaussian Processes

Density Function

If \( y \sim N(\mu, K) \), the probability density function is

\[
p(y; \mu, K) = \frac{1}{(2\pi)^{d/2} \det K} \exp \left( -\frac{1}{2} (y - \mu)^T K^{-1} (y - \mu) \right)
\]

This can also be called the likelihood function?

Therefore, a Gaussian process is uniquely defined by its mean and covariance functions

\[
\mu(x) = \mathbb{E} y_x, \quad \text{and} \quad K(x, z) = \text{cov}(y_x, y_z).
\]

We will write \( y_x \sim GP(\mu, K) \) for such a Gaussian process.

Question

What is the meaning of a singular \( K \) (Factor Analysis)?
Gaussian Processes: Goal

In practice, we assume that data \( \{ x_i, y_{x_i} \}^{N}_{i=1} \) is generated by a Gaussian process such that \( y \sim N(\mu, \Sigma) \).

Based on this, we may be interested in many things:

- Making predictions of unobserved values,
- Estimating the mean \( \mu(x) \) of the process, or
- Determining confidence for predictions.
In practice, we assume that data \( \{x_i, y_{x_i}\}_{i=1}^{N} \) is generated by a Gaussian process such that \( y \sim N(\mu, \Sigma) \).

Based on this, we may be interested in many things:

- Making predictions of unobserved values,
- Estimating the mean \( \mu(x) \) of the process, or
- Determining confidence for predictions.

**Question**

Are we interested in estimating the covariance of the process?
Approximation Theory

Your data \( \{x_i, y_i\}_{i=1}^N \) are the values of some function \( f \), possibly contaminated by noise. Your job is to reconstruct a good approximation \( s_f \) which can be used in place of \( f \).

- We assume that a deterministic function generated the data.
  - There may be some stochastic noise, but \( f \) is deterministic.
- Values in the convex hull of \( \{x_i\}_{i=1}^N \) can be interpolated using \( s_f \).
  - In the cardinal basis, these are a linear combination of the \( \{y_i\}_{i=1}^N \) values. Without the cardinal basis, we use \( K(\cdot, x_j) \) and \( c_j \), \( 1 \leq j \leq N \).
- Our interpolant \( s_f \) can serve as a **surrogate** for \( f \) in future computations, such as optimization or integration.
Gaussian Processes

Your data \( \{x_i, y_{x_i}\}_{i=1}^{N} \) are the values of one realization of \( y_x \sim GP(\mu, K) \). Your job is to use the values you are given to make predictions for \( y_{x_0} \) at as yet unobserved values \( x_0 \in \Omega \).

- We assume that a stochastic process generated the data.
  - There may be a deterministic component, as we will see later.
- Predictions can be made for any values in \( \Omega \).
  - The confidence with which we can make predictions is a function of (among other things) the design \( \{x_i\}_{i=1}^{N} \).
- The likelihood function gives a mechanism for determining “optimal” \( \mu \) and \( K \) functions.
Overlap Between Statistics and Analysis

Despite these different approaches, predicting unseen values $x$ may actually be identical for both Gaussian Processes and kernel interpolation.

Kernel Interpolation

Recall the interpolation system $Kc = y$, which defines the unique coefficients $c$ which interpolate the data for the chosen basis. Call

$$k(x)^T = (K(x, x_1), \cdots, K(x, x_N)).$$

If $K$ is invertible, then unknown $x$ can be interpolated as

$$s_f(x) = \sum_{j=1}^{N} c_j K(x, x_j) = k(x)^T c = k(x)^T K^{-1} y.$$

The situation for Gaussian Processes requires a lot more work.
Overlap Between Statistics and Analysis

To make predictions, we will consider two random vectors:

1. the vector $y$ at the points $\{x_i\}_{i=1}^N$, and
2. a (one element) vector $y_{x_0}$ which is our prediction at the point $x_0$.

Jointly, these have the normal distribution

$$
\begin{pmatrix}
  y \\
  y_{x_0}
\end{pmatrix}
\sim
\mathcal{N}
\left(
\begin{pmatrix}
  \mu \\
  \mu(x_0)
\end{pmatrix},
\begin{pmatrix}
  K & \mathbf{k}(x_0) \\
  \mathbf{k}(x_0)^T & K(x_0, x_0)
\end{pmatrix}
\right).
$$

Here we have assumed that $K$ is a symmetric kernel. We will denote the covariance matrix above as $\tilde{K}$.

Prediction

To make our prediction we need the distribution of $y_{x_0} | y$. 

Predictions for Gaussian Processes

With a whole lot of work (See Appendix A), we can show that

\[ y_{x_0} | y \sim N \left( \mu(x_0) + k(x_0)^T K^{-1}(y - \mu), K(x_0, x_0) - k(x_0)^T K^{-1} k(x_0) \right), \]

so the expected value is

\[ E y_{x_0} | y = \mu(x_0) + k(x_0)^T K^{-1}(y - \mu). \]

That is our prediction for future values \( x \) given \( (y_{x_i}, x_i)_{i=1}^N \).

Note

We will just use \( x \) rather than \( x_0 \) from now on. This is a bit ambiguous since \( y_x \) is how we denote the Gaussian Process as well. I need to think about how to handle this in general.
Matching the Predictions

For the kernel interpolation setting, our prediction for $x$ is

$$s_f(x) = k(x)^T K^{-1} y,$$

but for the Gaussian Processes setting we would predict

$$E y_x | y = \mu(x) + k(x)^T K^{-1}(y - \mu).$$

The Gaussian Process prediction looks much more complicated.
Matching the Predictions

For the kernel interpolation setting, our prediction for $x$ is

$$s_f(x) = k(x)^T K^{-1} y,$$

but for the Gaussian Processes setting we would predict

$$E y_x | y = \mu(x) + k(x)^T K^{-1} (y - \mu).$$

The Gaussian Process prediction looks much more complicated.

If we choose to restrict the class of Gaussian Processes we will consider to those with $\mu(x) \equiv 0$, then both predictions coincide.

**Question**

When we assume that $y_x \sim GP(0, K)$, both methods will yield the same predictions. Does that mean both methods are equivalent?
Distinct Differences

In what ways are these methods different, despite making the same predictions?

- **Kernel methods**
  - The interpolation points need not be equal to the kernel centers.
  - The kernels need not be positive definite.
  - Derivatives of $s_f$ can be computed directly.

- **Gaussian processes**
  - The (co)variance of our prediction is $K(x, x) - k(x)^T K^{-1} k(x)$.
  - We don’t need to assume that $\mu(x) = 0$.
  - The likelihood function $p$ helps quantify the role of $K$.

There is also a disconnect in terminology which can slow the movement of data between communities.
The variance of our (zero mean) prediction $y_x|y$ is $K(x, x) - k(x)^TK^{-1}k(x)$. This value comes from the assumption that $y$ and $y_x$ have a joint normal distribution.

This term is sometimes referred to as the **Kriging variance**, because the process of fitting spatial data is sometimes called Kriging.
Mismatched Jargon

The variance of our (zero mean) prediction $y_x | y$ is $K(x, x) - k(x)^T K^{-1} k(x)$. This value comes from the assumption that $y$ and $y_x$ have a joint normal distribution.

This term is sometimes referred to as the Kriging variance, because the process of fitting spatial data is sometimes called Kriging.

In a totally different setting, the power function is defined as

$$P(x) = \sqrt{K(x, x) - k(x)^T K^{-1} k(x)},$$

which has a relationship to the native space norm derived by $K$.

This value arises in native space norm error estimates for kernel interpolants.
Mismatched Jargon
Some of the most popular kernels are so-called Radial Basis Functions. These kernels satisfy

\[ K(x, z) = \phi(\|x - z\|). \]

Kernels of this type are easier to work with because, among other reasons, they are translation invariant.
Mismatched Jargon

Some of the most popular kernels are so-called Radial Basis Functions. These kernels satisfy

\[ K(x, z) = \phi(||x - z||). \]

Kernels of this type are easier to work with because, among other reasons, they are translation invariant.

Gaussian Process users may want similar properties for their covariance function:

\[ K(x + h, x) = \text{cov}(y_{x+h}, y_x) \]

is independent of \( x \), but the term often used is stationary. A stationary process must have constant mean.

In this setting, we may see the use of an autocovariance function:

\[ \gamma(h) = \text{cov}(y_{x+h}, y_x). \]

which means that \( y_x \sim N(\mu, \gamma(0)) \) for constant \( \mu \).
Recall the Hilbert-Schmidt SVD of the interpolation matrix $K$ as introduced by Dr. Fasshauer last week:

$$K = \psi \Lambda_1 \phi_1^T.$$ 

The $\Lambda_1$ matrix contains the first $N$ Hilbert-Schmidt eigenvalues in descending order and the $\psi$ and $\phi_1$ matrices are generated using the eigenfunctions.

This decomposition moves as much of the ill-conditioning as possible to the $\Lambda_1$ matrix.

Let’s take a look at how this plays a role in the stability of Gaussian Processes.
Hilbert-Schmidt SVD in Pieces
Recall the structure of the decomposition

\[ K = \Phi \left( \begin{array}{c} 1 \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{array} \right) \Lambda_1 \Phi_1^T \]

The data-dependence of the \( \psi_j \) basis is isolated to the “correction” term, meaning if we define

\[ \phi(x)^T = (\varphi_1(x), \cdots, \varphi_M(x)) \]

we can write our kernel basis using the HS SVD

\[ k(x)^T = \phi(x)^T \left( \begin{array}{c} 1 \\ \Lambda_2 \Phi_2^T \Phi_1^{-T} \Lambda_1^{-1} \end{array} \right) \Lambda_1 \Phi_1^T = \psi(x)^T \Lambda_1 \Phi_1^T, \]

where we have substituted in the stable basis

\[ \psi(x)^T = (\psi_1(x), \cdots, \psi_N(x)). \]
Computing Predictions

Our prediction for unobserved values of our Gaussian process is

$$E_y(x | y) = \mu(x) + k(x)^T K^{-1}(y - \mu).$$

Substituting the HS SVD gives

$$E_y(x | y) = \mu(x) + \psi(x)^T \Lambda_1 \Phi_1^T (\psi \Lambda_1 \Phi_1^T)^{-1} (y - \mu),$$

$$= \mu(x) + \psi(x)^T \Lambda_1 \Phi_1^T \Phi_1^{-T} \Lambda_1^{-1} \psi^{-1} (y - \mu),$$

$$= \mu(x) + \psi(x)^T \psi^{-1} (y - \mu),$$

which shows that by establishing the stable basis we can eliminate the ill-conditioning (caused by $\Lambda_1^{-1}$) in the predictions.
Likelihood functions for Gaussian Processes

As described earlier, the likelihood function associated with $y_x \sim GP(\mu, K)$ provides an opportunity to optimize (by some standard) the choice of covariance kernel.

Maximizing the likelihood function

$$p(y; \mu, K) = \frac{1}{\sqrt{(2\pi)^N \det K}} \exp \left( -\frac{1}{2} (y - \mu)^T K^{-1} (y - \mu) \right)$$

would find the kernel that most likely generated the data.
Likelihood functions for Gaussian Processes

Often times we want to maximize $p$ independent of any scaling of $K$.

In the interpolation setting, such a scaling $\sigma > 0$ of $K$ is immaterial, but for Gaussian Processes this scaling can affect the prediction variance, and in turn the likelihood.

The effect of the process variance on the likelihood is

$$p(y; \mu, \sigma K) = \frac{1}{\sqrt{(2\pi\sigma)^N \det K}} \exp \left( -\frac{1}{2\sigma} (y - \mu)^T K^{-1} (y - \mu) \right).$$
Likelihood functions for Gaussian Processes

In Appendix B, we substitute in the optimal value $\sigma_{opt}$ to find

$$p(y; \mu, \sigma_{opt}K) = \frac{1}{(2\pi)^{N/2}} \left[ (y - \mu)^T K^{-1} (y - \mu) \right]^{-N/2} \det(K)^{-1/2}$$

This function is subject to overflow and underflow. Instead we will take the logarithm and dispose of an additive and multiplicative factor (shown in the appendix) to produce

$$\tilde{p}(y; \mu, K) = \log \left( (y - \mu)^T K^{-1} (y - \mu) \right) + \frac{1}{N} \log \det K.$$

The kernel that maximizes the likelihood function $p$ is also the kernel that minimizes $\tilde{p}$. 

Mike McCourt
Gaussian Processes Q&A
Maximum Likelihood Estimation

We can use Maximum Likelihood Estimation to optimize our shape parameter. Minimizing

$$\tilde{\rho}(y; \mu, K) = \log \left( \left( y - \mu \right)^T K^{-1} (y - \mu) \right) + \frac{1}{N} \log \det K.$$ 

requires evaluating $\det(K)$, which, given its ill-conditioning, is a dicey proposition.

Using the Hilbert-Schmidt SVD this determinant can be computed in pieces

$$\det(K) = \det(\Psi \Lambda_1 \Phi_1) = \det(\Psi) \det(\Lambda_1) \det(\Phi_1)$$

The values $\det(\Psi)$ and $\det(\Phi_1)$ can be computed stably (we presume) using standard techniques.

Evaluating $\det(\Lambda_1)$ can be done analytically because it is a diagonal matrix populated by our Hilbert-Schmidt eigenvalues.
Maximum Likelihood Parameter Estimation

Of course, searching the entire space of all possible covariance kernels for the (possibly nonunique) optimal choice is a hopeless task.

A common practice (albeit not the only one) is to pick a family of kernels with a shape parameter $\varepsilon$, and try to determine the best possible $\varepsilon$.

We will demonstrate the benefit of the HS SVD on evaluating $\log \det K$ for a compact Matérn interpolant in 1D as $\varepsilon \to 0$ where $K$ becomes more ill-conditioned.
Stable Determinant Evaluation

Figure: $N = 30$ equally spaced points, $\beta = 8$
Some Problems Remain

Although we can be fairly confident in our computation of the determinant, there is another component of the $\tilde{p}$ function:

$$\log \left( (y - \mu)^T K^{-1} (y - \mu) \right).$$

Using the HS SVD does not really absolve us of our travails

$$\log \left( (y - \mu)^T \Phi_1^{-T} \Lambda_1^{-1} \Psi^{-1} (y - \mu) \right).$$

If we define the, hopefully, stably computable

$$y_\Psi = \Psi^{-1} (y - \mu)$$
$$y_\Phi = \Phi_1^{-1} (y - \mu)$$

then we can write

$$(y - \mu)^T \Phi_1^{-T} \Lambda_1^{-1} \Psi^{-1} (y - \mu) = y_\Phi^T \Lambda_1^{-1} y_\Psi,$$

and the ill-conditioning in the $\Lambda_1^{-1}$ remains.
Some Problems Remain

Although we haven’t completed this research yet, it is possible that working with the term

\[ y_\Phi^T \Lambda_1^{-1} y_\Psi, \]

is actually a tenable proposition.

We should be able to compute \( y_\Psi \) and \( y_\Phi \) without too much trouble if the \( \Psi \) basis and eigenfunction basis are reasonable choices to approximate \( y_x \).

If this is the case, then hopefully all the values of \( y_\Psi \) and \( y_\Phi \) should be roughly the same order, i.e.,

\[ \|y_\Psi\|_1 \approx N \|y_\Psi\|_\infty. \]

Under this assumption, only the final few values of \( y_\Phi^T \Lambda_1^{-1} y_\Psi \) are significant in the inner product. This may allow us to approximate it reasonably well.
LOOCV: How it works

Let \( s_f^{[k]} \) be the kernel interpolant to the training data \( \{ f_1, \ldots, f_{k-1}, f_{k+1}, \ldots, f_N \} \), i.e.,

\[
s_f^{[k]}(x) = \sum_{j=1}^{N} c_j^{[k]} K(x, x_j),
\]

such that

\[
s_f^{[k]}(x_i) = f_i, \quad i = 1, \ldots, k - 1, k + 1, \ldots, N,
\]
LOOCV: How it works

Let $s_f^{[k]}$ be the kernel interpolant to the training data $\{f_1, \ldots, f_{k-1}, f_{k+1}, \ldots, f_N\}$, i.e.,

$$s_f^{[k]}(x) = \sum_{j=1}^{N} \sum_{j \neq k} c_j^{[k]} K(x, x_j),$$

such that

$$s_f^{[k]}(x_i) = f_i, \quad i = 1, \ldots, k - 1, k + 1, \ldots, N,$$

and let $e_k(\varepsilon)$ be the error

$$e_k(\varepsilon) = f_k - s_f^{[k]}(x_k)$$

at the one validation point $x_k$ not used to determine the interpolant.
LOOCV: How it works

Let $s_f^{[k]}$ be the kernel interpolant to the training data \( \{ f_1, \ldots, f_{k-1}, f_{k+1}, \ldots, f_N \} \), i.e.,

$$s_f^{[k]}(x) = \sum_{j=1}^{N} \sum_{j \neq k} c_j^{[k]} K(x, x_j),$$

such that

$$s_f^{[k]}(x_i) = f_i, \quad i = 1, \ldots, k-1, k+1, \ldots, N,$$

and let $e_k(\varepsilon)$ be the error

$$e_k(\varepsilon) = f_k - s_f^{[k]}(x_k)$$

at the one validation point $x_k$ not used to determine the interpolant. Find

$$\varepsilon_{opt} = \arg\min_{\varepsilon} \|e(\varepsilon)\|, \quad e = [e_1, \ldots, e_N]^T$$
Efficient formula for the LOOCV criterion:

\[ e_k(\varepsilon) = \frac{c_k}{K_{kk}^{-1}}, \quad k = 1, \ldots, N, \quad (1) \]

with

- \( c_k \): \( k \)th coefficient of full interpolant \( s_f \)
- \( K_{kk}^{-1} \): \( k \)th diagonal element of inverse of corresponding interpolation matrix

was given by Rippa and Wahba.
Efficient formula for the LOOCV criterion:

\[ e_k(\varepsilon) = \frac{c_k}{K_{kk}^{-1}}, \quad k = 1, \ldots, N, \quad (1) \]

with

- \( c_k \): \( k^{th} \) coefficient of full interpolant \( s_f \)
- \( K_{kk}^{-1} \): \( k^{th} \) diagonal element of inverse of corresponding interpolation matrix

was given by Rippa and Wahba.

**Remark**

- Since both \( c_k \) and \( K_{kk}^{-1} \) need to be computed only once for each value of \( \varepsilon \) this results in \( \mathcal{O}(N^3) \) computational complexity.
Efficient formula for the LOOCV criterion:

\[ e_k(\varepsilon) = \frac{c_k}{K_{kk}^{-1}}, \quad k = 1, \ldots, N, \]  

(1)

with

- \( c_k \): \( k \)\textsuperscript{th} coefficient of full interpolant \( s_f \)
- \( K_{kk}^{-1} \): \( k \)\textsuperscript{th} diagonal element of inverse of corresponding interpolation matrix

was given by Rippa and Wahba.

**Remark**

- *Since both \( c_k \) and \( K_{kk}^{-1} \) need to be computed only once for each value of \( \varepsilon \) this results in \( \mathcal{O}(N^3) \) computational complexity.*

- *All entries in the error vector \( \mathbf{e} \) can be computed in a single statement in MATLAB if we vectorize the component formula (1):*

\[
\text{errorvector} = \frac{(\text{invIM*rhs})}{\text{diag(invIM)}};
\]
LOOCV via Hilbert-Schmidt SVD

Since the LOOCV criterion requires the inverse of $K$ (which may be very ill-conditioned), we use the Hilbert-Schmidt SVD

$$K = \Psi \Lambda \Phi^T,$$

accurate to within machine precision (i.e., truncation length $M > N$).
LOOCV via Hilbert-Schmidt SVD

Since the LOOCV criterion requires the inverse of $K$ (which may be very ill-conditioned), we use the Hilbert-Schmidt SVD

$$K = \Psi \Lambda_1 \Phi_1^T,$$

accurate to within machine precision (i.e., truncation length $M > N$). Then

$$K^{-1} = \Phi_1^{-T} \Lambda_1^{-1} \Psi^{-1}.$$
LOOCV via Hilbert-Schmidt SVD

Since the LOOCV criterion requires the inverse of $K$ (which may be very ill-conditioned), we use the Hilbert-Schmidt SVD

$$K = \Psi \Lambda_1 \Phi_1^T,$$

accurate to within machine precision (i.e., truncation length $M > N$). Then

$$K^{-1} = \Phi_1^{-T} \Lambda_1^{-1} \Psi^{-1}.$$

- The matrices $\Psi$ and $\Phi_1$ are usually “well-behaved”.
- But $\Lambda_1$ by itself still contains the basic ill-conditioning, i.e., potentially very small eigenvalues.
- We therefore use the pseudoinverse $\Lambda_1^\dagger$ instead of $\Lambda_1^{-1}$, i.e., we drop some of the smallest eigenvalues of the kernel $K$. 
LOOCV via Hilbert-Schmidt SVD

Since the LOOCV criterion requires the inverse of $K$ (which may be very ill-conditioned), we use the Hilbert-Schmidt SVD

$$K = \Psi \Lambda_1 \Phi_1^T,$$

accurate to within machine precision (i.e., truncation length $M > N$). Then

$$K^{-1} = \Phi_1^{-T} \Lambda_1^{-1} \Psi^{-1}.$$  

- The matrices $\Psi$ and $\Phi_1$ are usually “well-behaved”.
- But $\Lambda_1$ by itself still contains the basic ill-conditioning, i.e., potentially very small eigenvalues.
- We therefore use the pseudoinverse $\Lambda_1^\dagger$ instead of $\Lambda_1^{-1}$, i.e., we drop some of the smallest eigenvalues of the kernel $K$.

Remark

*Note that truncating the Hilbert-Schmidt SVD is fundamentally different from performing a standard SVD of $K$ and then truncating that.*
Example (Optimal $\varepsilon$-$\beta$ surfaces using MaternQR)

Determine the optimal $\varepsilon$ and $\beta$ for interpolation with compact Matérn kernels

\[ K_{\beta,\varepsilon}(x, z) = \sum_{n=1}^{\infty} 2 \left( n^2 \pi^2 + \varepsilon^2 \right)^{-\beta} \sin(n\pi x) \sin(n\pi z) \]

on $[0, 1]$ using $N = 24$ evenly spaced samples from

\[ f_\gamma(x) = \frac{1}{\left(\frac{1}{2} - \gamma\right)^2} \left( (x - \gamma)^2 + (1 - \gamma - x)^2 \right) e^{-36(x-0.4)^2}, \]

with $\gamma = 0.0567$. 
Optimal $\varepsilon$-$\beta$ surfaces using MaternQR (cont.)

- Error of true solution with $\varepsilon_{opt} = 18.047$, $\beta_{opt} = 4$ (left)
- Third out CV with $\varepsilon_{opt} = 14.251$, $\beta_{opt} = 6$ (right)
Optimal $\varepsilon$-$\beta$ surfaces using MaternQR (cont.)

- Half out CV with $\varepsilon_{\text{opt}} = 0.1$, $\beta_{\text{opt}} = 8$ (left)
- LOOCV with $\varepsilon_{\text{opt}} = 10.0$, $\beta_{\text{opt}} = 7$ (right).

For a smooth kernel (large $\beta$) we see instabilities for LOOCV for small $\varepsilon$ – even though the Hilbert-Schmidt SVD is used.
Summary

- Gaussian Processes are a useful way to describe scattered data interpolation problems.
- Some recent progress in kernel interpolation helps address problems in Gaussian Processes.
- We need to better consolidate knowledge from these two fields.

Other questions:
- What are Gaussian Linear Processes?
  - How does the Wold Decomposition play a part?
- What about Gaussian Processes in time?
- What is a best nonlinear predictor?
Appendix A: The Conditional Distribution $y_{x_0}|y$

Using the definition of conditional probability,

$$p(y_{x_0}|y; \mu, K)p(y; \mu, K) = p(y, y_{x_0}; \mu, K),$$

$$p(y_{x_0}|y; \mu, K) = \frac{1}{p(y; \mu, K)} p(y, y_{x_0}; \mu, K).$$

The joint probability density function is

$$p(y, y_{x_0}; \mu, K) = \frac{1}{(2\pi)^{d/2} \det \tilde{K}} \exp \left( -\frac{1}{2} \left( \begin{pmatrix} y \\ y_{x_0} \end{pmatrix} - \begin{pmatrix} \mu \\ \mu(x_0) \end{pmatrix} \right)^T \tilde{K}^{-1} \left( \begin{pmatrix} y \\ y_{x_0} \end{pmatrix} - \begin{pmatrix} \mu \\ \mu(x_0) \end{pmatrix} \right) \right).$$

Plugging this into the condition distribution gives

$$p(y_{x_0}|y; \mu, K) = Z_1 \exp \left( -\frac{1}{2} \left( \begin{pmatrix} y - \mu \\ y_{x_0} - \mu(x_0) \end{pmatrix} \right)^T \tilde{K}^{-1} \left( \begin{pmatrix} y - \mu \\ y_{x_0} - \mu(x_0) \end{pmatrix} \right) \right).$$

where $Z_1$ are the remaining terms that are omitted. It is a proportionality constant independent of $y_{x_0}$ that serves the purpose of normalizing the distribution.
Appendix A: The Conditional Distribution $y_{x_0} | y$

We need to understand the structure of $\tilde{K}^{-1}$ in terms of $K$. A quick study of the structure of a block matrix gives

$$\tilde{K}^{-1} = \begin{pmatrix} K & k(x_0) \\ k(x_0)^T & K(x_0, x_0) \end{pmatrix}^{-1} = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix},$$

where

$$A = (K - k(x_0)K(x_0, x_0)^{-1}k(x_0)^T)^{-1}$$
$$B = -(K - k(x_0)K(x_0, x_0)^{-1}k(x_0)^T)^{-1}k(x_0)K(x_0, x_0)^{-1}$$
$$C = (K(x_0, x_0) - k(x_0)^T K^{-1} k(x_0))^{-1}$$

All these inverses are guaranteed to exist if $K$ is a positive definite kernel.
Appendix A: The Conditional Distribution \( y_{x_0} | y \)

Using this with \( \tilde{K} \) allows us to compute

\[
\left( \begin{array}{c}
    y - \mu \\
    y_{x_0} - \mu(x_0)
\end{array} \right)^T \tilde{K}^{-1} \left( \begin{array}{c}
    y - \mu \\
    y_{x_0} - \mu(x_0)
\end{array} \right) =
\]

\[
(y - \mu)^T A (y - \mu) + (y - \mu)^T B (y_{x_0} - \mu(x_0)) +
\]

\[
(y_{x_0} - \mu(x_0))^T B^T (y - \mu) + (y_{x_0} - \mu(x_0))^T C (y_{x_0} - \mu(x_0)) =
\]

\[
(y - \mu)^T A (y - \mu) +
\]

\[
y^T B y_{x_0} - \mu^T B y_{x_0} - y^T B \mu(x_0) + \mu^T B \mu(x_0) +
\]

\[
y_{x_0}^T B^T y - \mu(x_0)^T B^T y - y_{x_0}^T B^T \mu + \mu(x_0)^T B^T \mu +
\]

\[
y_{x_0}^T C y_{x_0} - \mu(x_0)^T C y_{x_0} - y_{x_0}^T C \mu(x_0) + \mu(x_0)^T C \mu(x_0)
\]

Only the brown terms involve \( y_{x_0} \), so the magenta terms are consolidated into \( \hat{Z}_2 \). We will also use symmetry (i.e., \( y^T B y_{x_0} = y_{x_0}^T B^T y \)), which is necessary, although I’m not sure how to prove that it holds.
Appendix A: The Conditional Distribution $y_{x_0}|y$

Putting this result into $p(y_{x_0}|y)$ gives

$$p(y_{x_0}|y; \mu, K) = Z_1 \exp \left( -\frac{1}{2} \left( y_{x_0}^T C y_{x_0} - 2y_{x_0} C \mu(x_0) + y_{x_0}^T B^T (y - \mu) + \hat{Z}_2 \right) \right).$$

By defining

$$Z_2 = Z_1 e^{-\frac{1}{2} \hat{Z}_2},$$

we can write

$$p(y_{x_0}|y; \mu, K) = Z_2 \exp \left( -\frac{1}{2} \left( y_{x_0}^T C y_{x_0} - 2y_{x_0}^T C \mu(x_0) + 2y_{x_0}^T B^T (y - \mu) \right) \right).$$

Some matrix manipulations will make this manageable.
Appendix A: The Conditional Distribution $y_{x_0} | y$

Completing the square in a matrix sense means:

$$u^T Qu - 2u^T v = u^T Qu - 2u^T QQ^{-1} v$$
$$= u^T Qu - 2u^T QQ^{-1} v + v^T Q^{-1} v - v^T Q^{-1} v$$
$$= (u - Q^{-1} v)^T Q (u - Q^{-1} v) - v^T Q^{-1} v$$

where $Q$ is symmetric positive definite. Using this allows us to write

$$y_{x_0}^T C y_{x_0} - 2y_{x_0}^T \left( C \mu(x_0) - B^T (y - \mu) \right) =$$
$$\left( y_{x_0} - C^{-1} \left( C \mu(x_0) - B^T (y - \mu) \right) \right)^T C \left( y_{x_0} - C^{-1} \left( C \mu(x_0) - B^T (y - \mu) \right) \right)$$
$$- \left( C \mu(x_0) - B^T (y - \mu) \right)^T C^{-1} \left( C \mu(x_0) - B^T (y - \mu) \right)$$

Note that the second term related to $v^T Q^{-1} v$ has no $y_{x_0}$ terms in it.
Appendix A: The Conditional Distribution $y_{x_0} | y$

Let’s try and use what we know to clean this up some. First off, we replace the term without any $y_{x_0}$ with $\hat{Z}_3$:

$$\hat{Z}_3 = - \left( C\mu(x_0) - B^T (y - \mu) \right)^T C^{-1} \left( C\mu(x_0) - B^T (y - \mu) \right).$$

Defining

$$Z_3 = Z_2 e^{-\frac{1}{2} \hat{Z}_3}$$

allows us to write

$$p(y_{x_0} | y; \mu, K) = Z_3 \exp\left( - \frac{1}{2} \left( (y_{x_0} - \bar{\mu})^T C (y_{x_0} - \bar{\mu}) \right) \right),$$

where

$$\bar{\mu} = \mu(x_0) - C^{-1} B^T (y - \mu).$$
Appendix A: The Conditional Distribution $y_{x_0} | y$

With some work, we can simplify this expression for $\tilde{\mu}$. We can invoke the role of the inverse

$$\tilde{K}^{-1}\tilde{K} = I_{N+1}$$

$$\begin{pmatrix} A & B \\ B^T & C \end{pmatrix} \begin{pmatrix} K & k(x) \\ k(x)^T & K(x,x) \end{pmatrix} = \begin{pmatrix} I_N & I_1 \end{pmatrix}$$

The bottom left corner of the inverse requires

$$B^T K + C k(x)^T = 0$$

$$C^{-1}B^T = -k(x)^TK^{-1}$$

When plugged into $\tilde{\mu}$ we get

$$\tilde{\mu} = \mu(x_0) + k(x)^TK^{-1}(y - \mu).$$
Appendix A: The Conditional Distribution $y_{x_0} | y$

At this point, we can conclude that

$$p(y_{x_0} | y; \mu, K) = Z_3 \exp \left( -\frac{1}{2} \left( (y_{x_0} - \bar{\mu})^T C (y_{x_0} - \bar{\mu}) \right) \right),$$

which means that $y_{x_0} | y$ has a normal distribution

$$y_{x_0} | y \sim N(\bar{\mu}, C)$$

where

$$\bar{\mu} = \mu(x_0) + k(x)^T K^{-1} (y - \mu),$$

$$C = (K(x_0, x_0) - k(x_0)^T K^{-1} k(x_0))^{-1}$$
Appendix B: Considering the Process Variance

Recall that the likelihood of $y$ for a mean $\mu$ and covariance $\sigma K$ (where $\sigma > 0$ is the process variance) is

$$p(y; \mu, \sigma K) = \frac{1}{\sqrt{(2\pi\sigma)^N \det K}} \exp \left( -\frac{1}{2\sigma} (y - \mu)^T K^{-1} (y - \mu) \right).$$

I think that we can look at the profile likelihood, which is to say the marginal distribution

$$\int_0^\infty p(y; \mu, \sigma K) d\sigma$$

but I don’t remember how to do it. I’ll look at it some time.

Instead, we will determine the optimal value $\sigma_{opt}$ and use its value in our likelihood function.
Appendix B: Considering the Process Variance

We need to take the log to make this manageable,

\[ \log p = -\frac{N}{2} \log(2\pi \sigma) - \frac{1}{2} \log \det K - \frac{1}{2\sigma}(y - \mu)^T K^{-1}(y - \mu) \]

Differentiating gives

\[ \frac{d}{d\sigma} p(y; \mu, \sigma K) = -\frac{N}{2\sigma} + \frac{1}{2\sigma^2}(y - \mu)^T K^{-1}(y - \mu). \]

Setting this derivative equal to zero gives

\[ \sigma_{opt} = \frac{1}{N}(y - \mu)^T K^{-1}(y - \mu), \]

which is the optimal process variance.
Appendix B: Considering the Process Variance

Substituting in this optimal process variance gives

\[
p(y; \mu, \sigma^2_{opt}K) = \frac{1}{\sqrt{(2\pi)^{1/N}(y - \mu)^T K^{-1}(y - \mu))^N \det K}} e^{-N/2}.
\]

Taking the log gives

\[
\log p(y; \mu, \sigma^2_{opt}K) = -\frac{1}{2} \log \det K - \frac{N}{2} \left(1 + \log \left(\frac{2\pi}{N}(y - \mu)^T K^{-1}(y - \mu)\right)\right)
\]
\[
\propto -\frac{1}{2} \log \det K - \frac{N}{2} \log ((y - \mu)^T K^{-1}(y - \mu))
\]

We create the function \(\tilde{p}\) which we will minimize to maximize \(p\):

\[
\tilde{p}(y; \mu, K) = \log \det K + N \log ((y - \mu)^T K^{-1}(y - \mu)) .
\]