$\LaTeX$ command declarations here.

AERSP 597 - Machine Learning in Aerosapce Engineering

Lecture 10, Gaussian Process Regression: Probabilitic Formulation

Instructor: Daning Huang

In [2]:
```
from __future__ import division
from warnings import filterwarnings
filterwarnings('ignore')

import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

TODAY: Gaussian Process Regression - II

- Gaussian process
- Probabilistic formulation
- Computational aspects
- Determination of hyperparameters

References

- GPML Chps. 2, 4, 5
- DACE Toolbox Manual (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.3530&rep=rep1&type=pdf)

Gaussian Process

From wikipedia (https://en.wikipedia.org/wiki/Gaussian_process):

- A stochastic process is a collection of random variables indexed by time or space
- A Gaussian process (GP) is a stochastic process, such that every finite collection of those random variables has a **multivariate Gaussian** distribution

We say a function subjects to a GP,

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

where $m(\mathbf{x})$ is the mean function and $k(\mathbf{x}, \mathbf{x}')$ is the covariance function, when

$$m(\mathbf{x}) = \mathrm{E}[f(\mathbf{x})]$$
$$k(\mathbf{x}, \mathbf{x}') = \mathrm{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

The probabilistic distribution of a set of points $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ sampled from GP is Gaussian,

$$\mathbf{y} \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$$

where $[\mathbf{m}]_i = m(\mathbf{X}_i)$ and $[\mathbf{K}]_{ij} = k(\mathbf{X}_i, \mathbf{X}_j)$.

Covariance function

A covariance function of a Gaussian process is defined as,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 R(\mathbf{x}_i, \mathbf{x}_j)$$

where

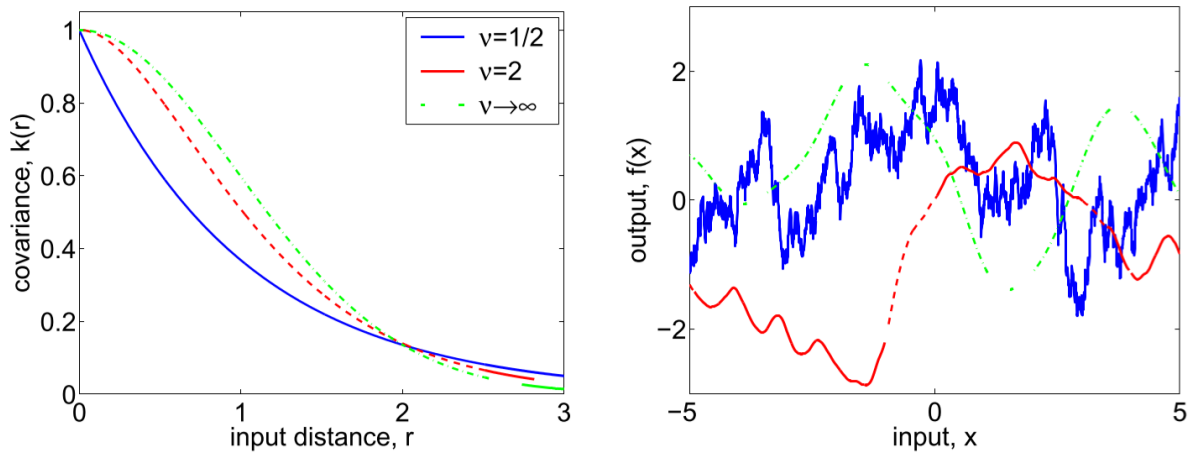
- σ_f^2 is the process variance
- R is typically stationary, i.e. $R(\mathbf{x}, \mathbf{x}') = R(\mathbf{x} - \mathbf{x}')$, and monotonically decreasing.
- Also assuming $R(0) = 1$, meaning that a point correlates with itself the most.

When the observation is noisy, the covariance function is modified as,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 R(\mathbf{x}_i, \mathbf{x}_j) + \sigma_n^2 \delta_{ij} \equiv \sigma_f^2 [R(\mathbf{x}_i, \mathbf{x}_j) + \tilde{\sigma}_n^2 \delta_{ij}]$$

Some possible covariance functions ($r = ||\mathbf{x} - \mathbf{x}'||$)

- Squared exponential (More discussion later)
$$k(r) = \exp\left[-\frac{r^2}{l^2}\right]$$
- Matérn class:
$$k(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} r}{l} \right)$$
- Piecewise polynomial, e.g.
$$k(r) = (1 - r)_+^j [(j + 1)r + 1]$$
where j depends on dimension, and it has a "compact support", i.e. non-zero within limited range.



(a) (b)

Figure 4.1: Panel (a): covariance functions, and (b): random functions drawn from Gaussian processes with Matérn covariance functions, eq. (4.14), for different values of ν, with $\ell = 1$. The sample functions on the right were obtained using a discretization of the x-axis of 2000 equally-spaced points.

A typical choice of R is the squared exponential function,

$$R(\mathbf{x}_i, \mathbf{x}_j) = \exp[-(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)]$$

where in the isotropic case, $\mathbf{M} = \frac{1}{2l^2}\mathbf{I}$ meaning that the length scales in all dimensions are the same; while in the anisotropic case, $\mathbf{M} = \frac{1}{2}diag(\mathbf{l})^{-2}$ is a diagonal matrix with positive diagonal entries, representing the length scales in different dimensions.

Hyperparameters so far:

- Length scales $\mathbf{l}$ - to be found by automatic relevance determination (https://www.slideshare.net/FlorianWilhelm2/explaining-the-idea-behind-automatic-relevance-determination-and-bayesian-interpolation-59498957) (ARD)
- Variances σ_f^2 and σ_n^2

The basic form

Return to the dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ sampled from GP:

$$\mathbf{y} \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$$

where $[\mathbf{m}]_i = m(\mathbf{X}_i)$ and $[\mathbf{K}]_{ij} = k(\mathbf{X}_i, \mathbf{X}_j)$. Note that scalar output is assumed.

Now split the points into two sets: the training set $\mathbf{X}_s$ with known output $\mathbf{y}_s$, and the target set $\mathbf{X}_u$ with unknown output $\mathbf{y}_s$. The joint distribution is still Gaussian, with the following mean,

$$\mathbf{m}^T = [\mathbf{m}(\mathbf{X}_s)^T, \mathbf{m}(\mathbf{X}_u)^T] \equiv [\mathbf{m}_s^T, \mathbf{m}_u^T]$$

and the block covariance matrix,

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{ss} & \mathbf{K}_{su} \\ \mathbf{K}_{us} & \mathbf{K}_{uu} \end{bmatrix}$$

where the i, jth element of $\mathbf{K}$ is associated with the ith and jth rows of $\mathbf{X}$,

$$[\mathbf{K}_{ab}]_{ij} = k([\mathbf{X}_a]_i, [\mathbf{X}_b]_j)$$

When $\mathbf{y}_s$ is noisy, a white noise term with variance σ_n^2 is added to the Gaussian distribution, adding a diagonal matrix to $\mathbf{K}_{ss}$

$$\mathbf{K}_y = \mathbf{K}_{ss} + \sigma_n^2 \mathbf{I}_{ss}$$

Prediction and error estimate

The distribution of $\mathbf{y}_u$ given $\mathbf{y}_s$ is found by conditional probability,

$$\mathbf{y}_u | \mathbf{y}_s \sim \mathcal{N}(\mathbf{m}_p, \mathbf{K}_p)$$

where the **predictive mean** is,

$$\mathbf{m}_p = \mathbf{m}_u + \mathbf{K}_{us}\mathbf{K}_y^{-1}(\mathbf{y}_s - \mathbf{m}_s)$$

and the **(predictive) covariance** is,

$$\mathbf{K}_p = \mathbf{K}_{uu} - \mathbf{K}_{us}\mathbf{K}_y^{-1}\mathbf{K}_{su}$$

What are the differences between the **predictive mean** and the **kernel ridge** formulation?

Using basis functions for the mean function

The form of mean function is usually unknown beforehand, so it is more common to use a set of basis functions,

$$m(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T b$$

The coefficients b have to be fitted from the sample data. Under a Bayesian framework, b is assumed to subject to,

$$b \sim \mathcal{N}(\mathbf{b}, \mathbf{B})$$

Taking b into account, the GP is modified as,

$$f(\mathbf{x}) \sim \mathcal{GP}(\mathbf{h}(\mathbf{x})^T\mathbf{b}, k(\mathbf{x}, \mathbf{x}') + \mathbf{h}(\mathbf{x})^T\mathbf{B}\mathbf{h}(\mathbf{x}))$$

Extra terms due to the basis functions shall be added to the covariance matrix,

$$\mathbf{K}_{ab} = \mathbf{K}(\mathbf{X}_a, \mathbf{X}_b) + \mathbf{H}_a\mathbf{B}\mathbf{H}_b^T, \quad [\mathbf{H}_*]_i = \mathbf{h}([\mathbf{X}_*]_i)^T$$

Simplification

The extra terms make the predictive mean and covariance from the previous section extremely cumbersome to compute. Simplifications are needed and enabled by invoking the [matrix inversion lemma (https://en.wikipedia.org/wiki/Woodbury_matrix_identity)](),

$$(\mathbf{A} + \mathbf{U}\mathbf{C}\mathbf{V}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^T\mathbf{A}^{-1}$$

In current case, the lemma is applied as follows,

$$(\mathbf{K}_y + \mathbf{H}\mathbf{B}\mathbf{H}^T)^{-1} = \mathbf{K}_y^{-1} - \mathbf{K}_y^{-1}\mathbf{H}\mathbf{C}^{-1}\mathbf{H}^T\mathbf{K}_y^{-1} \equiv \mathbf{K}_y^{-1} - \mathbf{A}$$

where $\mathbf{C} = \mathbf{B}^{-1} + \mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H}$ and the subscript s is dropped in $\mathbf{H}_s$. Furthermore, the following identities can be derived,

$$\mathbf{B}\mathbf{H}^T(\mathbf{K}_y^{-1} - \mathbf{A}) = \mathbf{C}^{-1}\mathbf{H}^T\mathbf{K}_y^{-1}$$

$$(\mathbf{K}_y^{-1} - \mathbf{A})\mathbf{H}\mathbf{B} = \mathbf{K}_y^{-1}\mathbf{H}\mathbf{C}^{-1}$$

The simplification procedure is bascally to cancel out matrices by combining the $\mathbf{H}$ and $(\mathbf{K}_y^{-1} - \mathbf{A})$ terms.

The predictive mean is simplified to,

$$\begin{aligned}\mathbf{m}_p^* &= \mathbf{H}_u\mathbf{b} + (\mathbf{K}_{us} + \mathbf{H}_u\mathbf{B}\mathbf{H}^T)(\mathbf{K}_y + \mathbf{H}\mathbf{B}\mathbf{H}^T)^{-1}(\mathbf{y}_s - \mathbf{H}\mathbf{b}) \\ &= \mathbf{K}_{us}\mathbf{K}_y^{-1}\mathbf{y}_s + \mathbf{D}^T\mathbf{C}^{-1}(\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{y}_s + \mathbf{B}^{-1}\mathbf{b})\end{aligned}$$

where $\mathbf{D} = \mathbf{H}_u^T - \mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{K}_{su}$. The covariance matrix is simplied to,

$$\begin{aligned}\mathbf{K}_p^* &= \mathbf{K}_{uu} + \mathbf{H}_u\mathbf{B}\mathbf{H}_u^T - (\mathbf{K}_{us} + \mathbf{H}_u\mathbf{B}\mathbf{H}^T)(\mathbf{K}_y + \mathbf{H}\mathbf{B}\mathbf{H}^T)^{-1}(\mathbf{K}_{su} + \mathbf{H}\mathbf{B}\mathbf{H}_u^T) \\ &= \mathbf{K}_p + \mathbf{D}^T\mathbf{C}^{-1}\mathbf{D}\end{aligned}$$

Finally, consider the case that the coefficients are distributed uniformly, instead of Gaussian. That means $\mathbf{b} \to 0$ and $\mathbf{B}^{-1} \to \mathbf{O}$.

With that, we arrive at the commonly used form of GPR,

$$\begin{aligned}\mathbf{m}_p^* &= \mathbf{K}_{us}\mathbf{K}_y^{-1}\mathbf{y}_s + \mathbf{D}^T(\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H})^{-1}\mathbf{H}\mathbf{K}_y^{-1}\mathbf{y}_s \\ &= \mathbf{K}_{us}\bar{\mathbf{g}} + \mathbf{H}_u\bar{\mathbf{b}} \\ \mathbf{K}_p^* &= \mathbf{K}_p + \mathbf{D}^T(\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H})^{-1}\mathbf{D}\end{aligned}$$

where $\bar{\mathbf{b}} = (\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{y}_s$ and $\bar{\mathbf{g}} = \mathbf{K}_y^{-1}(\mathbf{y}_s - \mathbf{H}\bar{\mathbf{b}})$. The term $\bar{\mathbf{b}}$ is essentially the coefficients of the mean function fitted from the sample data. Also, note that some people argue that the last term in $\mathbf{K}_p^*$ can neglected for simplicity [Sasena2002 (https://deepblue.lib.umich.edu/handle/2027.42/132844)].

Computational aspects

The computation of $\mathbf{m}_p^*$ and $\mathbf{K}_p^*$ can be tricky due to the ill-conditioned matrix inversion. The strategy is to combine cholesky decomposition (https://en.wikipedia.org/wiki/Cholesky_decomposition) and QR decomposition (https://en.wikipedia.org/wiki/QR_decomposition) to stabilize the inversions, i.e. $\mathbf{K}_y^{-1}$ and $(\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H})^{-1}$.

For inversion of $\mathbf{K}_y$,

$$\mathbf{K}_y^{-1}\mathbf{x} = (\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{x} = \mathbf{L}^{-T}(\mathbf{L}^{-1}\mathbf{x})$$

where $\mathbf{L}$ is a lower triangular matrix, and the matrix inversion is converted to two consecutive triangular solves.

To inverse $\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H}$,

$$\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H} = (\mathbf{L}^{-1}\mathbf{H})^T(\mathbf{L}^{-1}\mathbf{H}) \equiv \mathbf{F}^T\mathbf{F} = \mathbf{R}^T\mathbf{R}$$

where QR decomposition is used $\mathbf{Q}\mathbf{R} = \mathbf{F}$, $\mathbf{Q}$ is an orthogonal matrix, and $\mathbf{R}$ is an upper triangular matrix. $\mathbf{Q}$ is tall and slim, because the number of rows equals to the number of samples, while the number of columns equals to the number of basis functions.

After the stabilization, the quantities $\bar{\mathbf{b}}$ and $\bar{\mathbf{g}}$ in $\mathbf{m}_p^*$ are computed as follows,

$$\begin{aligned}\bar{\mathbf{b}} &= (\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{y}_s = \mathbf{R}^{-1}[\mathbf{Q}^T(\mathbf{L}^{-1}\mathbf{y}_s)] \\ \bar{\mathbf{g}} &= \mathbf{K}_y^{-1}(\mathbf{y}_s - \mathbf{H}\bar{\mathbf{b}}) = \mathbf{L}^{-T}[\mathbf{L}^{-1}(\mathbf{y}_s - \mathbf{H}\bar{\mathbf{b}})]\end{aligned}$$

The covariance matrix is computed as follows,

$$\mathbf{K}_p^* = \mathbf{K}_{uu} - (\mathbf{L}^{-1}\mathbf{K}_{su})^2 + (\mathbf{R}^{-T}[\mathbf{H}_u^T - \mathbf{F}^T(\mathbf{L}^{-1}\mathbf{K}_{su})])^2$$

where $(\square)^2 = \square^T \square$.

Are we done yet?

Yes...?

No, how about hyperparameters?

- Length scales $\mathbf{l}$ in the covariance functions
- Process variance σ_f^2
- Process noise σ_n^2
  - Determined by measurement
  - Zero if using computer simulations, but typical people use a small value for numerical stability

Preprocessing - Non-dimensionalization

For easier treatment, we non-dimensionalize some quantities in the GPR model.

The covariance matrices $\mathbf{K}_y$ and $\mathbf{K}_{su}$ can be "non-dimensionalized" by σ_f^2,

$$\mathbf{K}_y \equiv \sigma_f^2 \tilde{\mathbf{K}}_y \equiv \sigma_f^2(\tilde{\mathbf{K}}_{ss} + \tilde{\sigma}_n^2\mathbf{I})$$
$$\mathbf{K}_{su} \equiv \sigma_f^2 \tilde{\mathbf{K}}_{su}$$

Subsequently, for the predictive mean,

$$\mathbf{m}_p^* = \mathbf{K}_{su}^T\bar{\mathbf{g}} + \mathbf{H}_u^T\bar{\mathbf{b}} = \tilde{\mathbf{K}}_{su}^T\tilde{\mathbf{g}} + \mathbf{H}_u^T\tilde{\mathbf{b}}$$

where,

$$\bar{\mathbf{b}} = (\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{y}_s = (\mathbf{H}^T\tilde{\mathbf{K}}_y^{-1}\mathbf{H})^{-1}\mathbf{H}^T\tilde{\mathbf{K}}_y^{-1}\mathbf{y}_s \equiv \tilde{\mathbf{b}}$$
$$\bar{\mathbf{g}} = \mathbf{K}_y^{-1}(\mathbf{y}_s - \mathbf{H}\bar{\mathbf{b}}) = \sigma_f^{-2}\tilde{\mathbf{K}}_y^{-1}(\mathbf{y}_s - \mathbf{H}\tilde{\mathbf{b}}) \equiv \sigma_f^{-2}\tilde{\mathbf{g}}$$

And the covariance,

$$\begin{aligned}\mathbf{K}_p^* &= \mathbf{K}_{uu} - \mathbf{K}_{us}\mathbf{K}_y^{-1}\mathbf{K}_{su} + \mathbf{D}^T(\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H})^{-1}\mathbf{D} \\ &= \sigma_f^2[\tilde{\mathbf{K}}_{uu} - \tilde{\mathbf{K}}_{us}\tilde{\mathbf{K}}_y^{-1}\tilde{\mathbf{K}}_{su} + \tilde{\mathbf{D}}^T(\mathbf{H}^T\tilde{\mathbf{K}}_y^{-1}\mathbf{H})^{-1}\tilde{\mathbf{D}}]\end{aligned}$$

where,

$$\mathbf{D} = \mathbf{H}_u^T - \mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{K}_{su} = \mathbf{H}_u^T - \mathbf{H}^T\tilde{\mathbf{K}}_y^{-1}\tilde{\mathbf{K}}_{su} \equiv \tilde{\mathbf{D}}$$

In sum, σ_f^2 has no direct effect on the mean, but the covariance is proportional to σ_f^2, once the length scales are given.

Learning the hyperparameters

Log-likelihood

The hyperparameters are determined using the maximum likelihood estimation (MLE). With the joint Gaussian distribution, the log marginal likelihood of the training data is,

$$\begin{aligned}\mathcal{L} &= \log p(\mathbf{y}_s|\mathbf{X}_s, \mathbf{b}, \mathbf{B}) \\ &= -\frac{1}{2}(\mathbf{y}_s - \mathbf{Hb})^T(\mathbf{K}_y + \mathbf{H}^T\mathbf{BH})^{-1}(\mathbf{y}_s - \mathbf{Hb}) - \frac{1}{2}\log|\mathbf{K}_y + \mathbf{H}^T\mathbf{BH}| - \frac{n}{2}\log 2\pi\end{aligned}$$

where n is number of training data points.

Utilizing the determinant counterpart of matrix inversion lemma,

$$|\mathbf{A} + \mathbf{UCV}^T| = |\mathbf{A}||\mathbf{C}||\mathbf{C}^{-1} + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{U}|$$

and let $\mathbf{b} = 0$ and $\mathbf{B}^{-1} \to \mathbf{O}$ as was done last time,

$$-2\mathcal{L} = \mathbf{y}_s^T[\mathbf{K}_y^{-1} - \mathbf{K}_y^{-1}\mathbf{H}(\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{K}_y^{-1}]\mathbf{y}_s + \log|\mathbf{K}_y| + \log|\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H}| + (n-m)\log 2$$

where m is number of basis functions, and was introduced due to the singularity caused by $|\mathbf{B}|$.

An interesting simplification can be done to the first term in the above expression,

$$\begin{aligned}I_1 &= \mathbf{y}_s^T[\mathbf{K}_y^{-1} - \mathbf{K}_y^{-1}\mathbf{H}(\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{K}_y^{-1}]\mathbf{y}_s \\ &= \mathbf{y}_s^T\mathbf{K}_y^{-1}(\mathbf{y}_s - \mathbf{H}\bar{\mathbf{b}}) \equiv I_2 \\ I_1 &= (\mathbf{y}_s - \mathbf{H}\bar{\mathbf{b}})^T\mathbf{K}_y^{-1}\mathbf{y}_s \equiv I_3 \\ I_1 &= \mathbf{y}_s^T\mathbf{K}_y^{-1}\mathbf{y}_s - \bar{\mathbf{b}}^T\mathbf{H}^T\mathbf{K}_y^{-1}\mathbf{H}\bar{\mathbf{b}} \equiv I_4 \\ I_1 &= I_2 + I_3 - I_4 = (\mathbf{y}_s - \mathbf{H}\bar{\mathbf{b}})^T\mathbf{K}_y^{-1}(\mathbf{y}_s - \mathbf{H}\bar{\mathbf{b}})\end{aligned}$$

Indicating that the term I_1 behaves as if it is centered at $\mathbf{H}\bar{\mathbf{b}}$, with a covariance matrix $\mathbf{K}_y$.

Factor $\mathcal{L}$ with σ_f^2,

$$-2\tilde{\mathcal{L}}_1 = \sigma_f^{-2}(\mathbf{y}_s - \mathbf{H}\tilde{\mathbf{b}})^T\tilde{\mathbf{K}}_y^{-1}(\mathbf{y}_s - \mathbf{H}\tilde{\mathbf{b}}) + \log|\tilde{\mathbf{K}}_y| + \log|\mathbf{H}^T\tilde{\mathbf{K}}_y^{-1}\mathbf{H}| + (n-m)\log\sigma_f^2$$

where the constant term is neglected.

If the prior on the coefficients of the basis functions is ignored, the log-likelihood simplifies to,

$$-2\tilde{\mathcal{L}}_2 = \sigma_f^{-2}(\mathbf{y}_s - \mathbf{H}\tilde{\mathbf{b}})^T\tilde{\mathbf{K}}_y^{-1}(\mathbf{y}_s - \mathbf{H}\tilde{\mathbf{b}}) + \log|\tilde{\mathbf{K}}_y| + n\log\sigma_f^2$$

Process variances

A natural step next would be finding σ_f^2 by setting $\partial(-2\tilde{\mathcal{L}})/\partial\sigma_f^2 = 0$ and solving for σ_f^2,

$$\sigma_f^2 = \frac{1}{N}(\mathbf{y}_s - \mathbf{H}\tilde{\mathbf{b}})^T\tilde{\mathbf{K}}_y^{-1}(\mathbf{y}_s - \mathbf{H}\tilde{\mathbf{b}})$$

where $N = n - m$ for $\tilde{\mathcal{L}}_1$, and $N = n$ for $\tilde{\mathcal{L}}_2$. The former case is the center estimate of the variance, while the latter the MLE. When there are *multiple* outputs, the output variables are usually assumed to be independent, and σ_f^2 can be computed separately for each output.

Plug the value back to the likelihood, and remove the constant terms [Welch1992 (http://www.tandfonline.com/doi/abs/10.1080/00401706.1992.10485229)], one obtains reduced log likelihood,

$$\begin{aligned}-\mathcal{F}_1 &= \log|\tilde{\mathbf{K}}_y| + \log|\mathbf{H}^T\tilde{\mathbf{K}}_y^{-1}\mathbf{H}| + (n-m)\log\sigma_f^2 \\ -\mathcal{F}_2 &= \log|\tilde{\mathbf{K}}_y| + n\log\sigma_f^2\end{aligned}$$

where the only remaining unknown hyperparameters are the length scales $\mathbf{l}$. Note that $\mathcal{F}_2$ should be used if a general mean function, without priors on its coefficients, is employed.

The minimization of $-\mathcal{F}_2$ is equivalent to the minimization of

$$-\mathcal{F}^* = \sigma_f^2 |\tilde{\mathbf{K}}_y|^{1/n}$$

where $|\tilde{\mathbf{K}}_y| = |\tilde{\mathbf{L}}|^2$ using Cholesky decomposition $\tilde{\mathbf{K}}_y = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^T$.

Finding the hyperparameters

- Two-step method
  - Utilizes the reduced log likelihood, $\mathcal{F}_2$
  - Primary unknown: length scales $\mathbf{l}$
  - Dependent hyperparameters: σ_f^2 and σ_n^2
- One-step method
  - Determine all hyperparameters, even including $\bar{\mathbf{b}}$, simultaneously
  - Typically relies on gradient-based algorithms
  - Efficient if implemented under a differentiable programming framework

> In the homework you will implement the two-step method