

TRAINING OF A GAUSSIAN PROCESS

● To make predictions for a regression task by a Gaussian process, one simply needs to perform matrix computations.

Gaussian Process Training

- But for this to work out, we assume that the covariance functions is fully given, including all of its hyperparameters.



- A very nice property of GPs is that we can learn the numerical hyperparameters of a selected covariance function directly during GP training.



Learning goals

- Training of GPs via Maximum Likelihood estimation of its hyperparameters
- Computational complexity is governed by matrix inversion of the covariance matrix

TRAINING A GP VIA MAXIMUM LIKELIHOOD

Let us assume

- To make predictions for a regression task by a Gaussian process, one simply needs to perform matrix computations.

$$\mathbf{y} = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2),$$

where $f(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, \text{kk}(\mathbf{x}, \mathbf{x}' | \theta))$. Assume that the covariance function is fully given, including all of its hyperparameters.

Observing $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathbf{I})$, the marginal log-likelihood (or evidence) is

- A very nice property of GPs is that we can learn the numerical hyperparameters of a selected covariance function directly during

$$\begin{aligned} \log p(\mathbf{y} | \mathbf{X}, \theta) &= \log \left[(2\pi)^{-n/2} |\mathbf{K}_y|^{-1/2} \exp \left(-\frac{1}{2} \mathbf{y}^\top \mathbf{K}_y^{-1} \mathbf{y} \right) \right] \\ &= -\frac{1}{2} \mathbf{y}^\top \mathbf{K}_y^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_y| - \frac{n}{2} \log 2\pi. \end{aligned}$$

with $\mathbf{K}_y := \mathbf{K} + \sigma^2 \mathbf{I}$ and θ denoting the hyperparameters (the parameters of the covariance function).



TRAINING A GP VIA MAXIMUM LIKELIHOOD / 2

The three terms of the marginal likelihood have interpretable roles, considering that the model becomes less flexible as the length-scale increases:

$$y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2),$$

where $f(\mathbf{x}) \sim \mathcal{GP}(\mu, k(\mathbf{x}, \mathbf{x} | \theta))$.

- the data fit $-\frac{1}{2} \mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y}$, which tends to decrease if the length scale increases

Observing $\mathbf{y} | \mathbf{x}, \theta$ with σ^2 fixed, the marginal log-likelihood (evidence) is

- the complexity penalty $-\frac{1}{2} \log |\mathbf{K}_y|$, which depends on the covariance function only and which increases with the length-scale, because the model gets less complex with growing length-scale

$$\log p(\mathbf{y} | \mathbf{x}, \theta) = \log \left[(2\pi)^{-n/2} |\mathbf{K}_y|^{-1/2} \exp \left(-\frac{1}{2} \mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y} \right) \right]$$

- a normalization constant $-\frac{n}{2} \log 2\pi$

$$= -\frac{1}{2} \mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_y| - \frac{n}{2} \log 2\pi.$$

with $\mathbf{K}_y := \mathbf{K} + \sigma^2 \mathbf{I}$ and θ denoting the hyperparameters (the parameters of the covariance function).



TRAINING A GP: EXAMPLE UM LIKELIHOOD

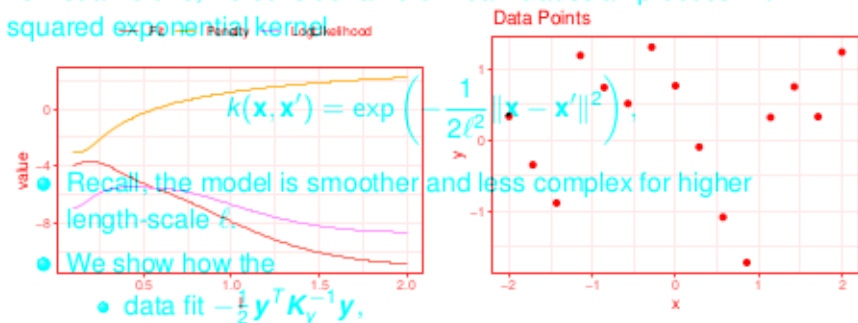
To visualize this, we consider a zero-mean Gaussian process with, squared exponential kernel becomes less flexible as the length-scale increases:

- the data fit $-\frac{1}{2}(\mathbf{y}, \mathbf{K}_y)^{-1} \mathbf{y}$, which tends to increase if the length scale increases
- Recall, the model is smoother and less complex for higher length-scale ℓ .
the complexity penalty $-\frac{1}{2} \log |\mathbf{K}_y|$, which depends on the length-scale function only and which increases with the length-scale
- We show how the
 - data fit $-\frac{1}{2} \mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y}$,
 - a normalization constant $-\frac{n}{2} \log 2\pi - \frac{1}{2} \log |\mathbf{K}_y|$, and
 - the overall value of the marginal likelihood $\log p(\mathbf{y} | \mathbf{X}, \theta)$behave for increasing value of ℓ .



TRAINING A GP: EXAMPLE / 2

To visualize this, we consider a zero-mean Gaussian process with squared exponential kernel



- Recall, the model is smoother and less complex for higher length-scale ℓ .

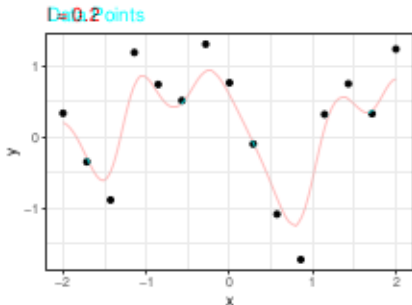
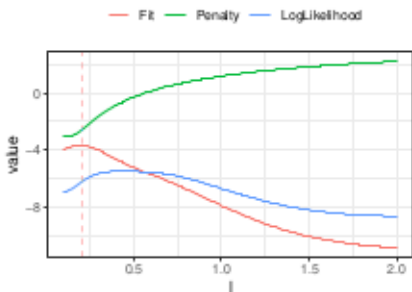
- We show how the

- data fit $-\frac{1}{2} \mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y}$,
- the complexity penalty $-\frac{1}{2} \log |\mathbf{K}_y|$, and

The left plot shows how values of the data fit $-\frac{1}{2} \mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y}$, the complexity penalty $-\frac{1}{2} \log |\mathbf{K}_y|$ (high value means less penalization) and the overall marginal likelihood $\log p(\mathbf{y} | \mathbf{X}; \theta)$ behave for increasing values of ℓ .

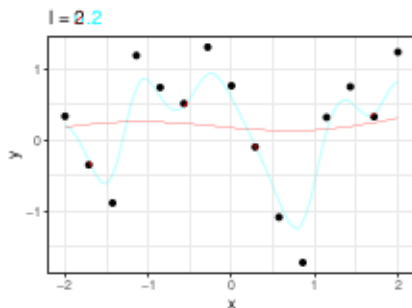
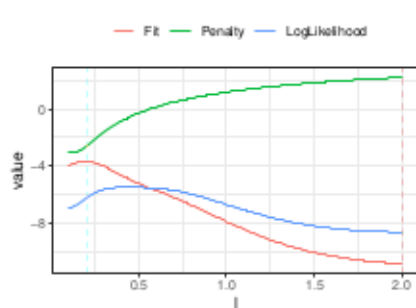


TRAINING A GP: EXAMPLE / 3



The left plot shows how values of the data fit $= \frac{1}{2} \mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y}$, the complexity penalty $-\frac{1}{2} \log |\mathbf{K}_y|$ (high value means less penalization) and the overall marginal likelihood $\log p(\mathbf{y} | \mathbf{X}, \theta)$ behave for increasing values of ℓ .
A small ℓ results in a good fit, but a high complexity penalty (low $-\frac{1}{2} \log |\mathbf{K}_y|$).

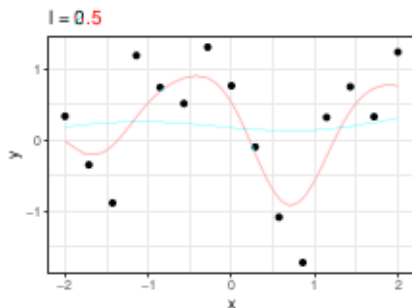
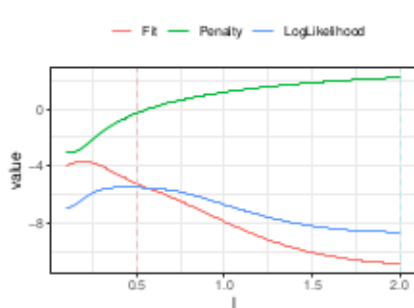
TRAINING A GP: EXAMPLE / 4



The left plot shows how values of the data fit $-\frac{1}{2}\mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y}$, the complexity penalty $-\frac{1}{2} \log |\mathbf{K}_y|$ (high value means less penalization) and the overall marginal likelihood $\log p(\mathbf{y} | \mathbf{X}, \theta)$ behave for increasing values of l .

A large l results in a poor fit, but a high complexity penalty (low $-\frac{1}{2} \log |\mathbf{K}_y|$).

TRAINING A GP: EXAMPLE / 5

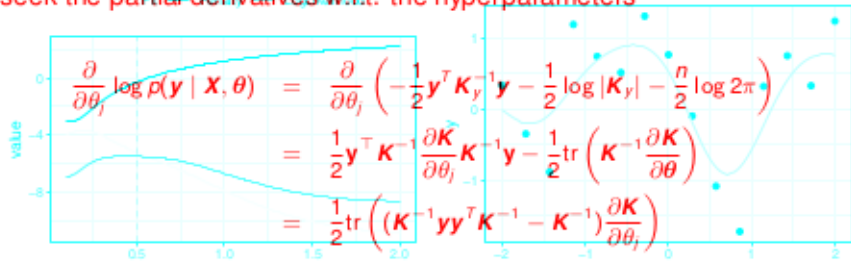


The left plot shows how values of the data fit $-\frac{1}{2}\mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y}$, the complexity penalty $-\frac{1}{2} \log |\mathbf{K}_y|$ (high value means less penalization) and the overall marginal likelihood $\log p(\mathbf{y} | \mathbf{X}, \theta)$ behave for increasing values of ℓ .

The maximizer of the log-likelihood, $\ell = 0.5$, balances complexity and fit.

TRAINING A GP: VIA MAXIMUM LIKELIHOOD

To set the hyperparameters by maximizing the marginal likelihood, we seek the partial derivatives w.r.t. the hyperparameters



using $\frac{\partial}{\partial \theta_j} \mathbf{K}^{-1} = -\mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \theta_j} \mathbf{K}^{-1}$ and $\frac{\partial}{\partial \theta} \log |\mathbf{K}| = \text{tr} \left(\mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \theta} \right)$.

The left plot shows how values of the data fit $-\frac{1}{2} \mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y}$, the complexity penalty $-\frac{1}{2} \log |\mathbf{K}_y|$ (high value means less penalization) and the overall marginal likelihood $\log p(\mathbf{y} | \mathbf{X}, \theta)$ behave for increasing values of ℓ .

The maximizer of the log-likelihood, $\ell = 0.5$, balances complexity and fit.



TRAINING A GP VIA MAXIMUM LIKELIHOOD / 2

The complexity and the runtime of training a Gaussian process is dominated by the computational task of inverting \mathbf{K} - or let's rather say for decomposing it.

- Standard methods require $\mathcal{O}(n^3)$ time (!) for this.
- Once \mathbf{K}^{-1} - or rather the decomposition - is known, the computation of the partial derivatives requires only $\mathcal{O}(n^2)$ time per hyperparameter.
- Thus, the computational overhead of computing derivatives is small, so using a gradient based optimizer is advantageous.

using $\frac{\partial}{\partial \theta_j} \mathbf{K}^{-1} = -\mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \theta_j} \mathbf{K}^{-1}$ and $\frac{\partial}{\partial \theta} \log |\mathbf{K}| = \text{tr} \left(\mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \theta} \right)$.



TRAINING A GP VIA MAXIMUM LIKELIHOOD / 3

Workarounds to make GP estimation feasible for big data include:

- using kernels that yield sparse \mathbf{K} : cheaper to invert.
 - subsampling the data to estimate θ : $\mathcal{O}(m^3)$ for subset of size m .
 - Standard methods require $\mathcal{O}(n^3)$ time (!) for this.
 - combining estimates on different subsets of size m :
 - **Bayesian committee**, $\mathcal{O}(nm^2)$.
 - computation of the partial derivatives requires only $\mathcal{O}(n^2)$ time per hyperparameter.
 - using low-rank approximations of \mathbf{K} by using only a representative subset ("inducing points") of m training data \mathbf{X}_m :
 - **Nystrom approximation** $\mathbf{K} \approx \mathbf{K}_{nm} \mathbf{K}_{mm}^{-1} \mathbf{K}_{mn}$,
 $\mathcal{O}(nmk + m^3)$ for a rank- k approximate inverse of \mathbf{K}_{mm} .
 - Thus, the computational overhead of computing derivatives is small, so using a gradient based optimizer is advantageous.
 - exploiting structure in \mathbf{K} induced by the kernel: exact solutions but complicated maths, not applicable for all kernels.
- ... this is still an active area of research.



TRAINING A GP VIA MAXIMUM LIKELIHOOD

Workarounds to make GP estimation feasible for big data include:

- using kernels that yield sparse \mathbf{K} : cheaper to invert.
- subsampling the data to estimate θ : $\mathcal{O}(m^3)$ for subset of size m .
- combining estimates on different subsets of size m :
Bayesian committee, $\mathcal{O}(nm^2)$.
- using low-rank approximations of \mathbf{K} by using only a representative subset ("inducing points") of m training data \mathbf{X}_m :
Nyström approximation $\mathbf{K} \approx \mathbf{K}_{nm} \mathbf{K}_{mm}^{-1} \mathbf{K}_{mn}$,
 $\mathcal{O}(nmk + m^3)$ for a rank- k -approximate inverse of \mathbf{K}_{mm} .
- exploiting structure in \mathbf{K} induced by the kernel: exact solutions but complicated maths, not applicable for all kernels.

... this is still an active area of research.