Gaussian Processes

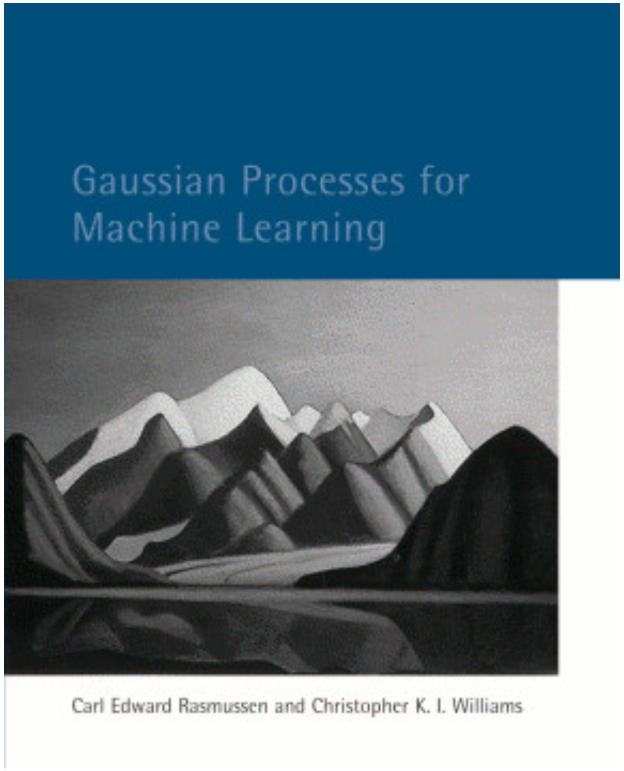
Marc Deisenroth
Centre for Artificial Intelligence
Department of Computer Science
University College London

@mpd37
`m.deisenroth@ucl.ac.uk`
`https://deisenroth.cc`

AIMS Rwanda and AIMS Ghana

March/April 2020

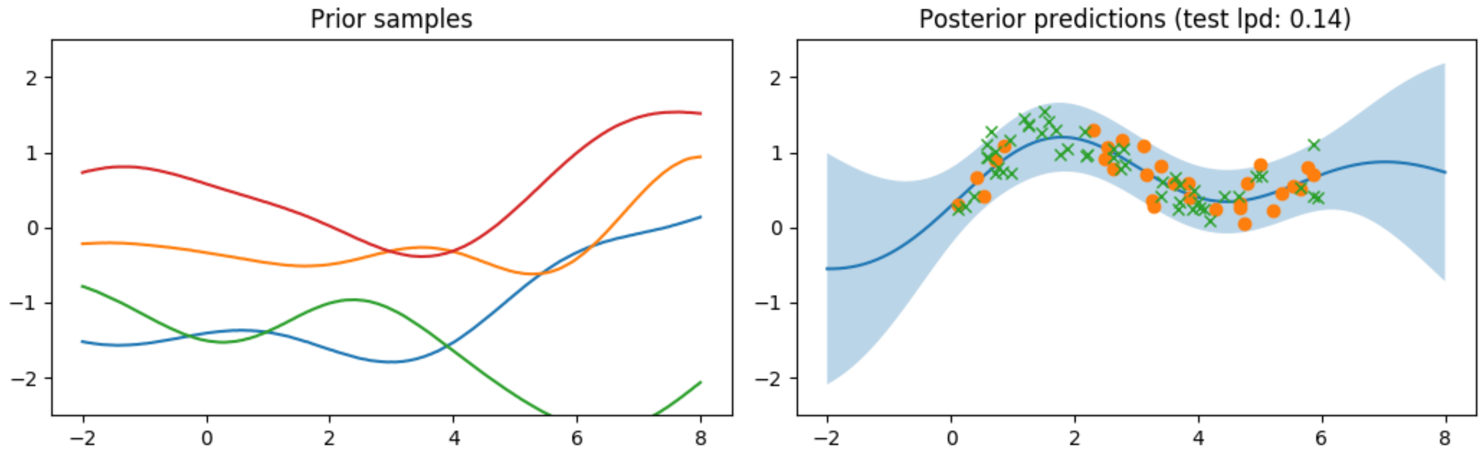
Reference


Gaussian Processes for Machine Learning
Carl Edward Rasmussen and Christopher K. I. Williams

`http://www.gaussianprocess.org/`

Model Selection

Influence of Prior on Posterior

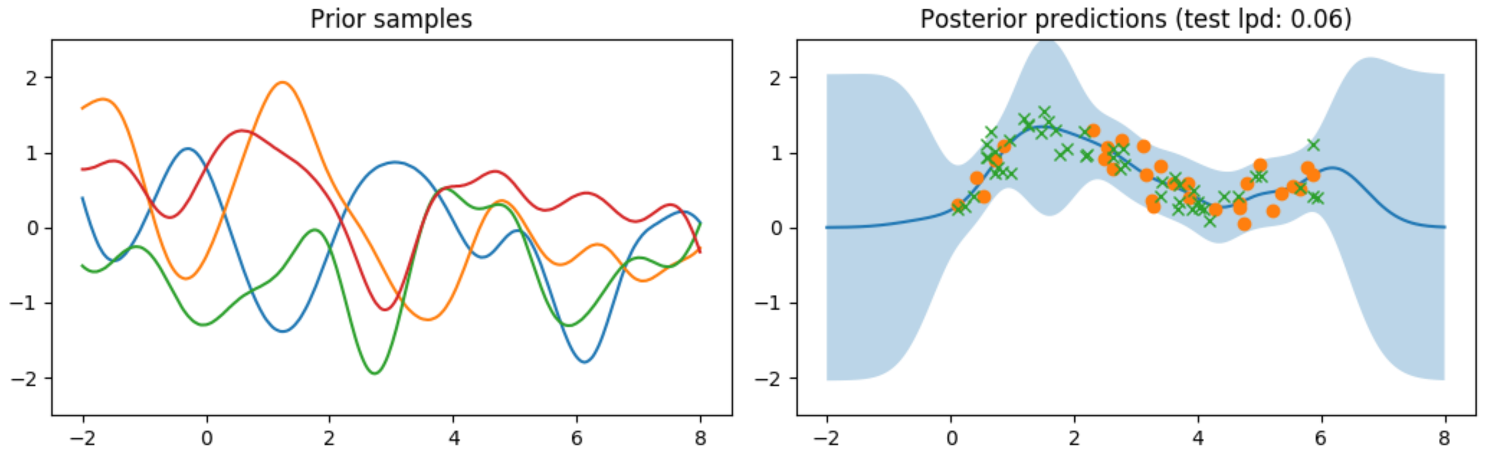


- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_* | \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}, \ell)$$

for different length-scales ℓ and different datasets

Influence of Prior on Posterior



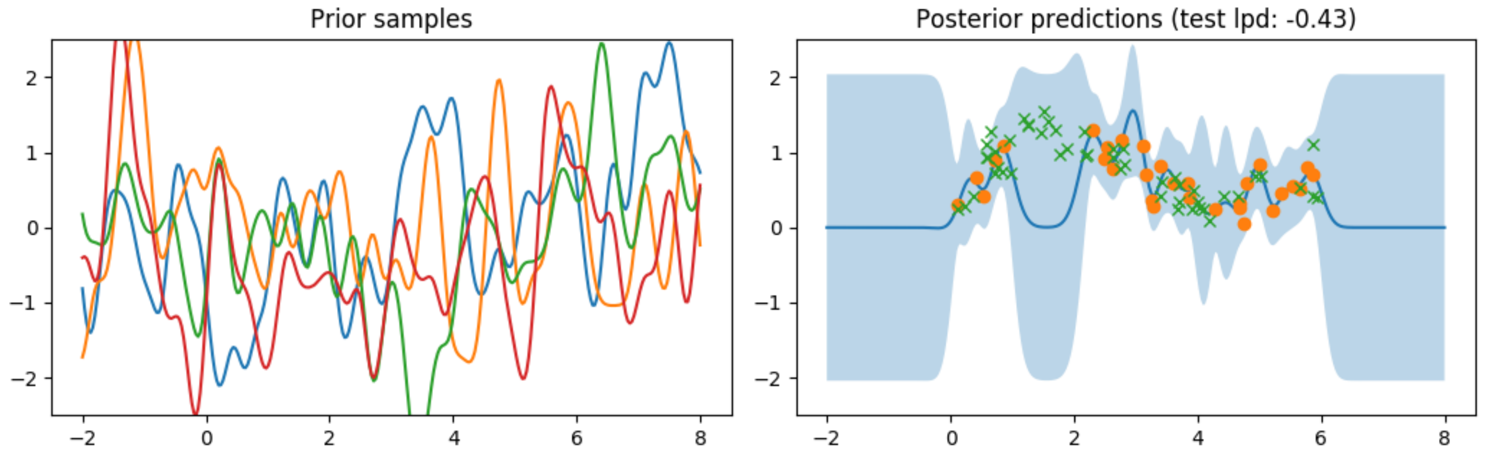
- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}, \ell)$$

for different length-scales ℓ and different datasets

- Shorter length-scale ⏩ More flexible model ⏩ Faster increase in uncertainty away from data ⏩ Bad generalization properties

Influence of Prior on Posterior



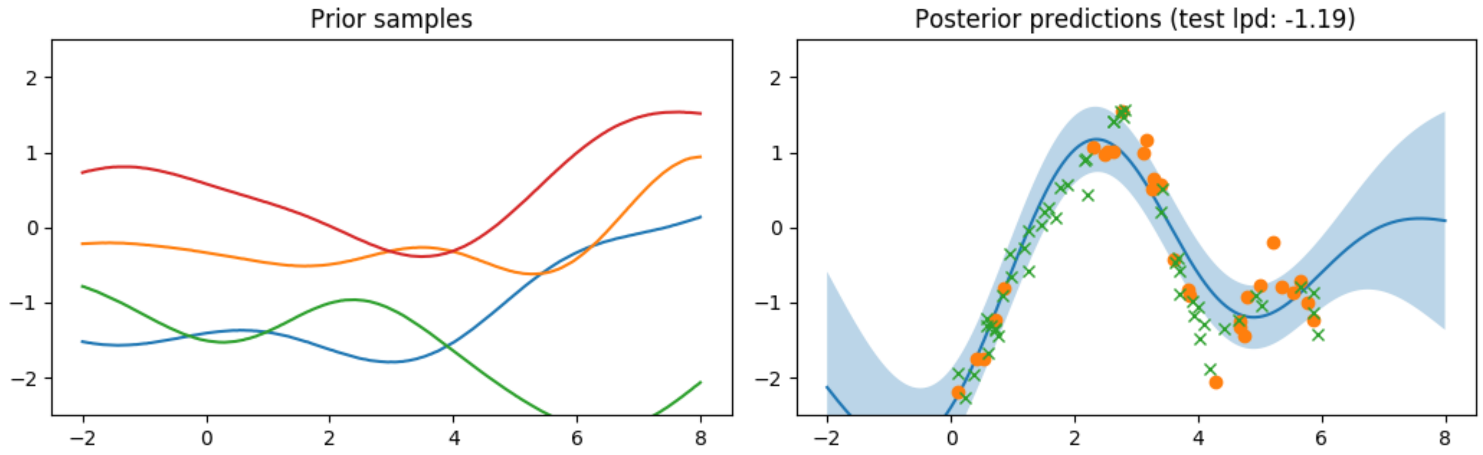
- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_* | \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}, \ell)$$

  for different length-scales ℓ and different datasets

- Shorter length-scale ⏩ More flexible model ⏩ Faster increase in uncertainty away from data ⏩ Bad generalization properties

Influence of Prior on Posterior



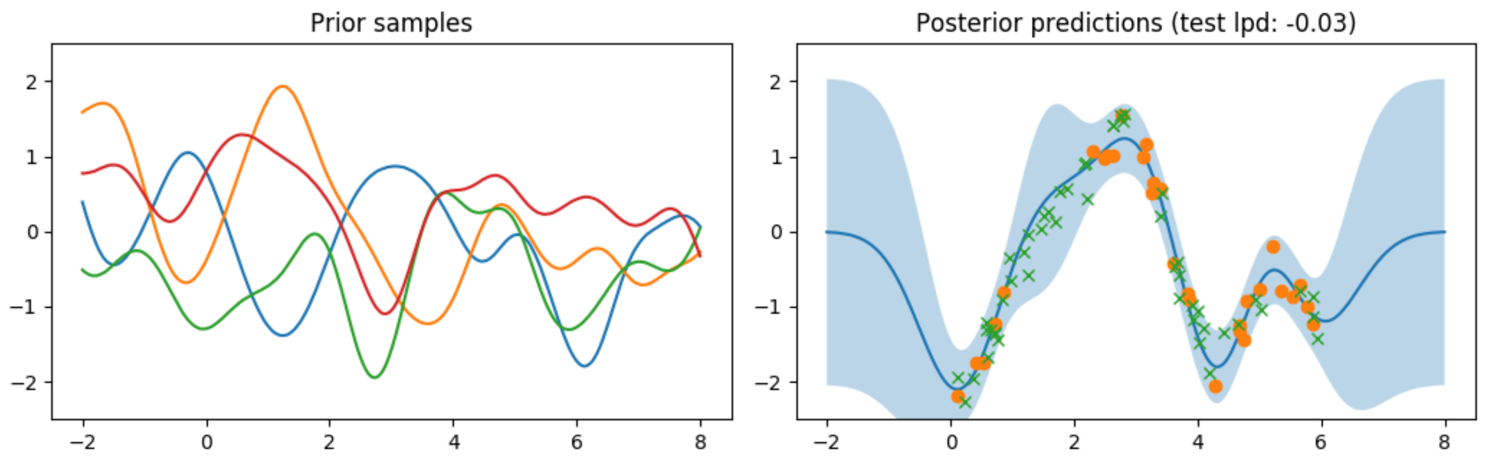
- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}, \ell)$$

for different length-scales ℓ and different datasets

- Shorter length-scale ⏩ More flexible model ⏩ Faster increase in uncertainty away from data ⏩ Bad generalization properties

Influence of Prior on Posterior



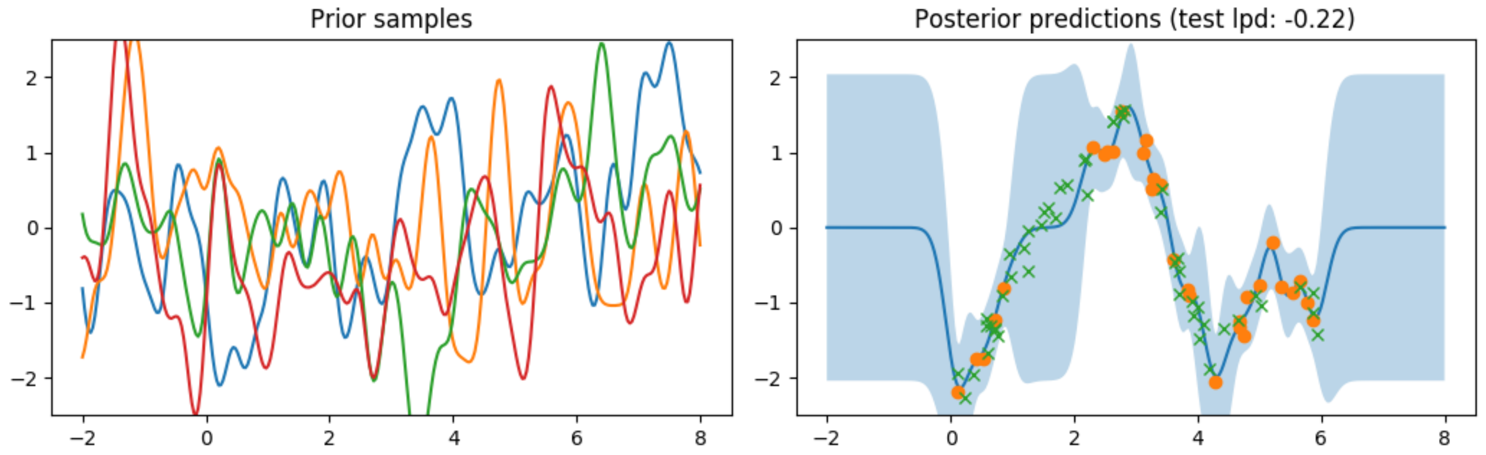
- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}, \ell)$$

  for different length-scales ℓ and different datasets

- Shorter length-scale ⏩ More flexible model ⏩ Faster increase in uncertainty away from data ⏩ Bad generalization properties

Influence of Prior on Posterior

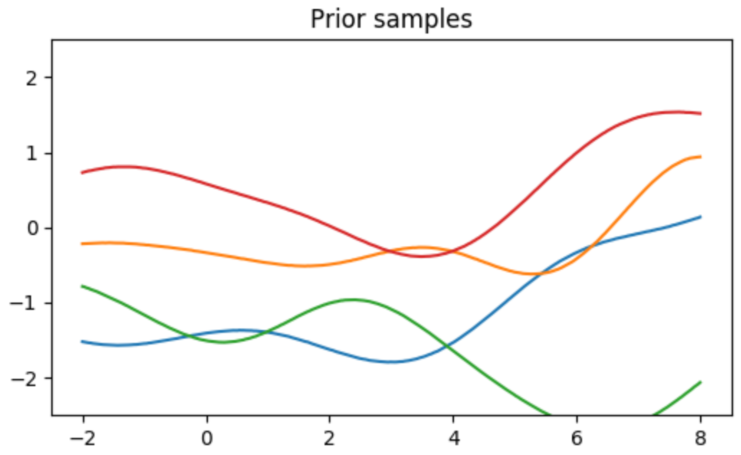


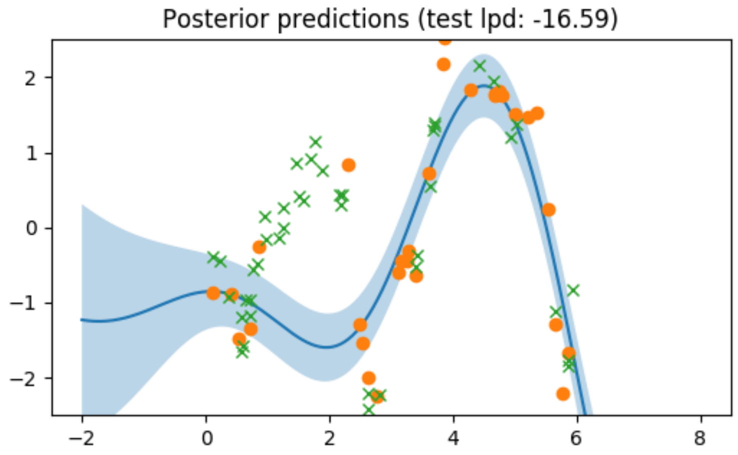
- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_* | \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}, \ell)$$

  for different length-scales ℓ and different datasets
- Shorter length-scale ⏩ More flexible model ⏩ Faster increase in uncertainty away from data ⏩ Bad generalization properties

Influence of Prior on Posterior



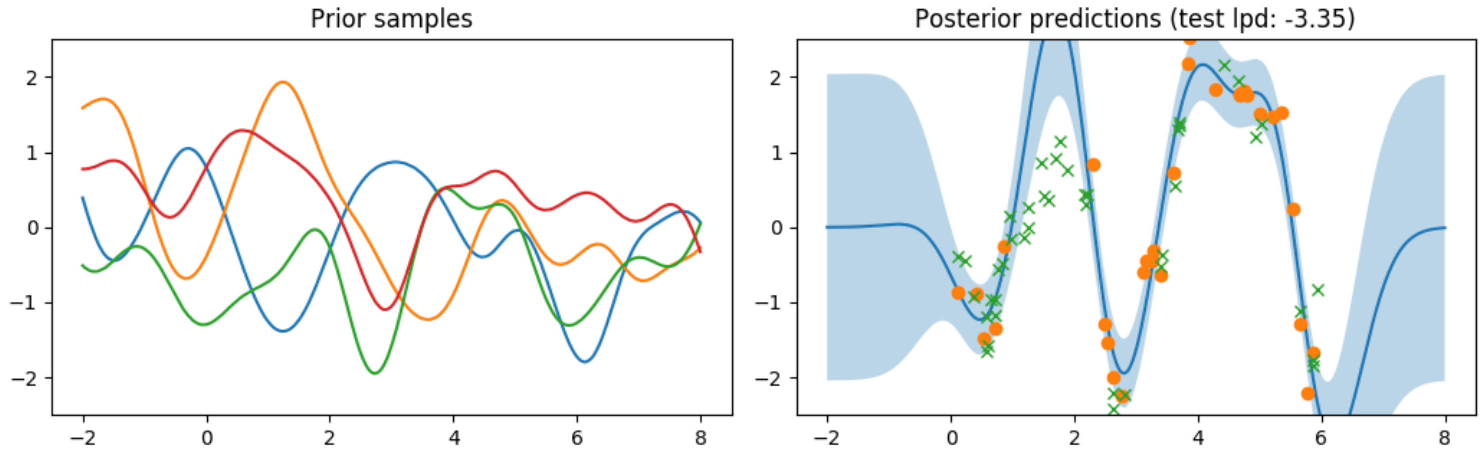


- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}, \ell)$$

  for different length-scales ℓ and different datasets
- Shorter length-scale ⏩ More flexible model ⏩ Faster increase in uncertainty away from data ⏩ Bad generalization properties

Influence of Prior on Posterior

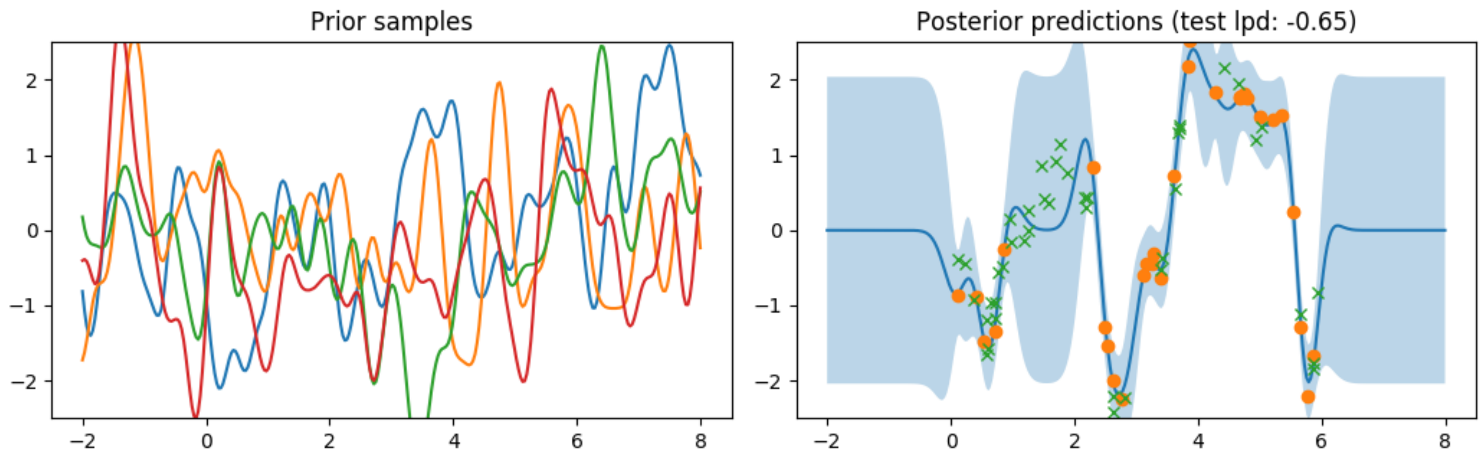


- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_* | \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}, \ell)$$

  for different length-scales ℓ and different datasets
- Shorter length-scale ⏩ More flexible model ⏩ Faster increase in uncertainty away from data ⏩ Bad generalization properties

Influence of Prior on Posterior



- Generalization error measured by log-predictive density (lpd)

$$\text{lpd} = \log p(y_* | \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}, \ell)$$

  for different length-scales ℓ and different datasets

- Shorter length-scale ⏩ More flexible model ⏩ Faster increase in uncertainty away from data ⏩ Bad generalization properties

Model Selection

- Make predictions equipped with uncertainty
- Choice of prior (e.g., length-scale) influences predictions
- Different tasks require different priors

Model Selection

- Make predictions equipped with uncertainty
- Choice of prior (e.g., length-scale) influences predictions
- Different tasks require different priors

How do we select a good prior?

Model Selection

- Make predictions equipped with uncertainty
- Choice of prior (e.g., length-scale) influences predictions
- Different tasks require different priors

How do we select a good prior?

Model Selection in GPs

- Choose hyper-parameters of the GP
- Choose good mean function and kernel

Hyper-Parameters of a GP

The GP possesses a set of hyper-parameters:

- Parameters of the mean function
- Parameters of the covariance function (e.g., length-scales and signal variance)
- Likelihood parameters (e.g., noise variance σ_n^2)

Hyper-Parameters of a GP

The GP possesses a set of hyper-parameters:

- Parameters of the mean function
- Parameters of the covariance function (e.g., length-scales and signal variance)
- Likelihood parameters (e.g., noise variance σ_n^2)

▶▶ Train a GP to find a good set of hyper-parameters

Hyper-Parameters of a GP

The GP possesses a set of hyper-parameters:

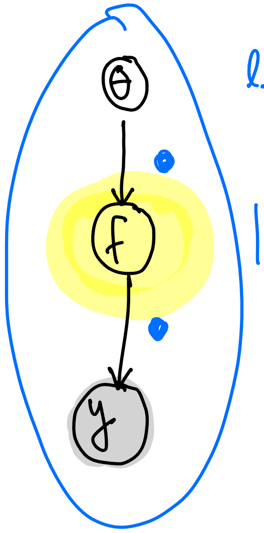
- Parameters of the mean function
- Parameters of the covariance function (e.g., length-scales and signal variance)
- Likelihood parameters (e.g., noise variance σ_n^2)

▸▸ Train a GP to find a good set of hyper-parameters

▸▸ Higher-level model selection to find good mean and covariance functions
(can also be automated: Automatic Statistician (Lloyd et al., 2014))

GP

$p(a) = \int p(a,b)\,db$ sum rule



level 2 hyper-parameters of the Gaussian process

level 1 unobserved (latent) function

observed function values (noisy)

$$\underset{\theta}{\operatorname{argmax}} \boxed{p(y|\theta)}$$

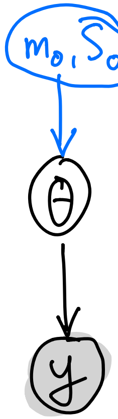
sum rule $\hookrightarrow \int p(\underline{y,f}|\theta)\,df$

$$= \underline{\int p(y|f)\,p(f|\theta)\,df} = p(y|\theta)$$

likelihood GP prior

This is the marginal likelihood (Maximum likelihood Type 2)

Linear Regression



level 1

param. of the model

- Maximum likelihood
- MAP

$\underset{\theta}{\operatorname{argmax}}\ p(y|\theta)$

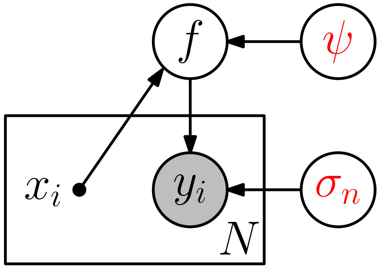
$\underset{\theta}{\operatorname{argmax}}\ p(\theta|y)$

Prior on θ + Bayes' theorem

Gaussian Process Training: Hyper-Parameters

GP Training

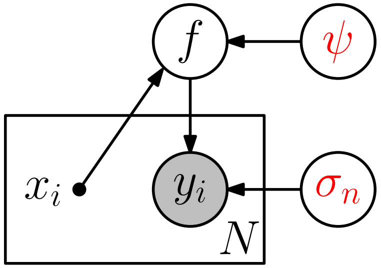
Find good hyper-parameters $\boldsymbol{\theta}$ (kernel/mean function parameters ψ, noise variance σ_n^2)



Gaussian Process Training: Hyper-Parameters

GP Training

Find good hyper-parameters $\boldsymbol{\theta}$ (kernel/mean function parameters ψ, noise variance σ_n^2)



- Place a prior $p(\boldsymbol{\theta})$ on hyper-parameters
- Posterior over hyper-parameters:

$$p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{\theta})\, p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{X})}$$

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \int p(\boldsymbol{y}|f, \boldsymbol{X})p(f|\boldsymbol{X}, \boldsymbol{\theta})\mathrm{d}f$$

Gaussian Process Training: Hyper-Parameters

marginal likelihood = evidence

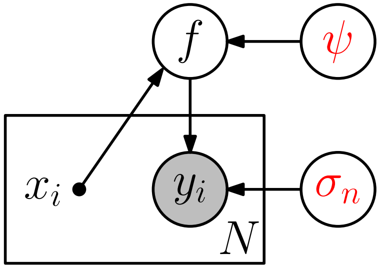
= marginalized likelihood

- Posterior over hyper-parameters: "integrate out"

$$p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \frac{p(\boldsymbol{\theta})\, p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{X})}$$

likelihood

$$p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = \int p(\boldsymbol{y}|f(\boldsymbol{X}))\, p(f(\boldsymbol{X})|\boldsymbol{\theta})\, \mathrm{d}f$$



$$p(y|x) = \int p(y|x,\theta)\, p(\theta)\, d\theta$$

$\longrightarrow$ marginalized marginal likelihood

marginal likelihood

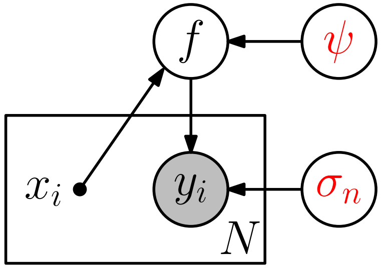
$$= \int \underbrace{\int p(y|f(x))\, p(f(x)|\theta)\, df}_{\text{marginal likelihood = evidence}}\; p(\theta)\, d\theta$$

Gaussian Process Training: Hyper-Parameters

- Posterior over hyper-parameters:

$$p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \frac{p(\boldsymbol{\theta})\, p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{X})}$$

$$p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = \int p(\boldsymbol{y}|f(\boldsymbol{X}))\, p(f(\boldsymbol{X})|\boldsymbol{\theta})\, \mathrm{d}f$$



- Choose hyper-parameters $\boldsymbol{\theta}^*$, such that

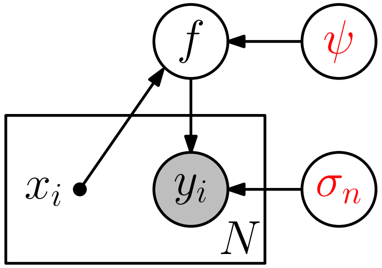
$$\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})$$

Gaussian Process Training: Hyper-Parameters

- Posterior over hyper-parameters:

$$p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \frac{p(\boldsymbol{\theta})\,p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{X})}$$

$$p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = \int p(\boldsymbol{y}|f(\boldsymbol{X}))\,p(f(\boldsymbol{X})|\boldsymbol{\theta})\,\mathrm{d}f$$



- Choose hyper-parameters $\boldsymbol{\theta}^*$, such that

$$\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})$$

▸▸ Maximize marginal likelihood if $p(\boldsymbol{\theta}) = \mathcal{U}$ (uniform prior)

Training via Marginal Likelihood Maximization

GP Training

Maximize the evidence/marginal likelihood (probability of the data given the hyper-parameters, where the unwieldy f has been integrated out) ⏩ Also called Maximum Likelihood Type-II

Training via Marginal Likelihood Maximization

GP Training

Maximize the evidence/marginal likelihood (probability of the data given the hyper-parameters, where the unwieldy f has been integrated out) ⏩ Also called Maximum Likelihood Type-II

Marginal likelihood (with a prior mean function $m(\cdot) \equiv 0$):

$$
\begin{aligned}
p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) &= \int p(\boldsymbol{y}|f(\boldsymbol{X}))\, p(f(\boldsymbol{X})|\boldsymbol{\theta})\, \mathrm{d}f \\
&= \int \mathcal{N}\big(\boldsymbol{y} \,|\, f(\boldsymbol{X}),\, \sigma_n^2 \boldsymbol{I}\big)\, \mathcal{N}\big(f(\boldsymbol{X}) \,|\, \boldsymbol{0},\, \boldsymbol{K}\big)\, \mathrm{d}f \\
&= \mathcal{N}\big(\boldsymbol{y} \,|\, \boldsymbol{0},\, \boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\big)
\end{aligned}
$$

Training via Marginal Likelihood Maximization

GP Training

Maximize the evidence/marginal likelihood (probability of the data given the hyper-parameters, where the unwieldy f has been integrated out) ⏩ Also called Maximum Likelihood Type-II

Marginal likelihood (with a prior mean function $m(\cdot) \equiv 0$):

$$
\begin{aligned}
p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) &= \int p(\boldsymbol{y}|f(\boldsymbol{X}))\, p(f(\boldsymbol{X})|\boldsymbol{\theta})\,\mathrm{d}f \\
&= \int \mathcal{N}\big(\boldsymbol{y}\,|\,f(\boldsymbol{X}),\,\sigma_n^2\boldsymbol{I}\big)\,\mathcal{N}\big(f(\boldsymbol{X})\,|\,\boldsymbol{0},\,\boldsymbol{K}\big)\,\mathrm{d}f \\
&= \mathcal{N}\big(\boldsymbol{y}\,|\,\boldsymbol{0},\,\boldsymbol{K}+\sigma_n^2\boldsymbol{I}\big)
\end{aligned}
$$

Learning the GP hyper-parameters:

$$
\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})
$$

Gradient-based Optimization

- Log-marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\tfrac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\boldsymbol{y} - \tfrac{1}{2}\log|\boldsymbol{K}_{\boldsymbol{\theta}}| + \text{const}$$

$$\boldsymbol{K}_{\boldsymbol{\theta}} := \boldsymbol{K} + \sigma_n^2\boldsymbol{I}$$

Gradient-based Optimization

- Log-marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = -\tfrac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\boldsymbol{y} - \tfrac{1}{2}\log|\boldsymbol{K}_{\boldsymbol{\theta}}| + \text{const}$$

$$\boldsymbol{K}_{\boldsymbol{\theta}} := \boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$$

- Gradient-based optimization to get hyper-parameters $\boldsymbol{\theta}^*$:

$$\begin{aligned}\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})}{\partial \theta_i} &= \tfrac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\frac{\partial \boldsymbol{K}_{\boldsymbol{\theta}}}{\partial \theta_i}\boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\boldsymbol{y} - \tfrac{1}{2}\text{tr}\big(\boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\frac{\partial \boldsymbol{K}_{\boldsymbol{\theta}}}{\partial \theta_i}\big)\\ &= \tfrac{1}{2}\text{tr}\big((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \boldsymbol{K}_{\boldsymbol{\theta}}^{-1})\frac{\partial \boldsymbol{K}_{\boldsymbol{\theta}}}{\partial \theta_i}\big)\,,\end{aligned}$$

$$\boldsymbol{\alpha} := \boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\boldsymbol{y}$$

Marginal Likelihood Illustration[1]

- “ELBO” refers to the log-marginal likelihood
- Data-fit term gets worse, but marginal likelihood increases

[1] Thanks to Mark van der Wilk

Inspecting the Marginal Likelihood

Log-marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = -\tfrac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\boldsymbol{y} - \tfrac{1}{2}\log|\boldsymbol{K}_{\boldsymbol{\theta}}| + \text{const}\,, \qquad \boldsymbol{K}_{\boldsymbol{\theta}} := \boldsymbol{K} + \sigma_n^2\boldsymbol{I}$$

Inspecting the Marginal Likelihood

Log-marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = -\tfrac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\boldsymbol{y} - \tfrac{1}{2}\log|\boldsymbol{K}_{\boldsymbol{\theta}}| + \text{const}\,, \quad \boldsymbol{K}_{\boldsymbol{\theta}} := \boldsymbol{K} + \sigma_n^2\boldsymbol{I}$$

- Quadratic term measures whether observation $\boldsymbol{y}$ is within the variation allowed by the prior

Inspecting the Marginal Likelihood

Log-marginal likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = -\tfrac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\boldsymbol{y} - \tfrac{1}{2}\log|\boldsymbol{K}_{\boldsymbol{\theta}}| + \text{const}\,, \qquad \boldsymbol{K}_{\boldsymbol{\theta}} := \boldsymbol{K} + \sigma_n^2\boldsymbol{I}$$

- Quadratic term measures whether observation $\boldsymbol{y}$ is within the variation allowed by the prior
- Determinant is the product of the variances of the prior (volume of the prior) ⏩ Volume $\approx$ richness of model class

Inspecting the Marginal Likelihood

Log-marginal likelihood:

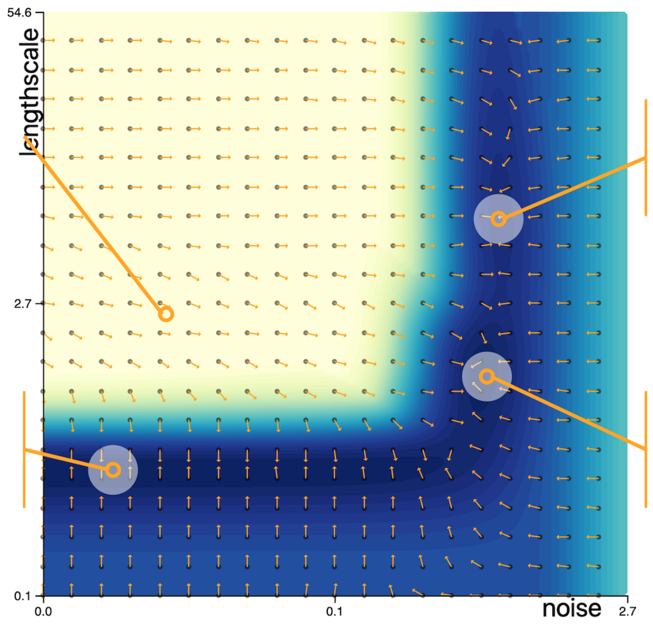
$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = -\tfrac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\boldsymbol{y} - \tfrac{1}{2}\log|\boldsymbol{K}_{\boldsymbol{\theta}}| + \text{const}\,, \qquad \boldsymbol{K}_{\boldsymbol{\theta}} := \boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$$

- Quadratic term measures whether observation $\boldsymbol{y}$ is within the variation allowed by the prior
- Determinant is the product of the variances of the prior (volume of the prior) ▶▶ Volume $\approx$ richness of model class

Marginal likelihood

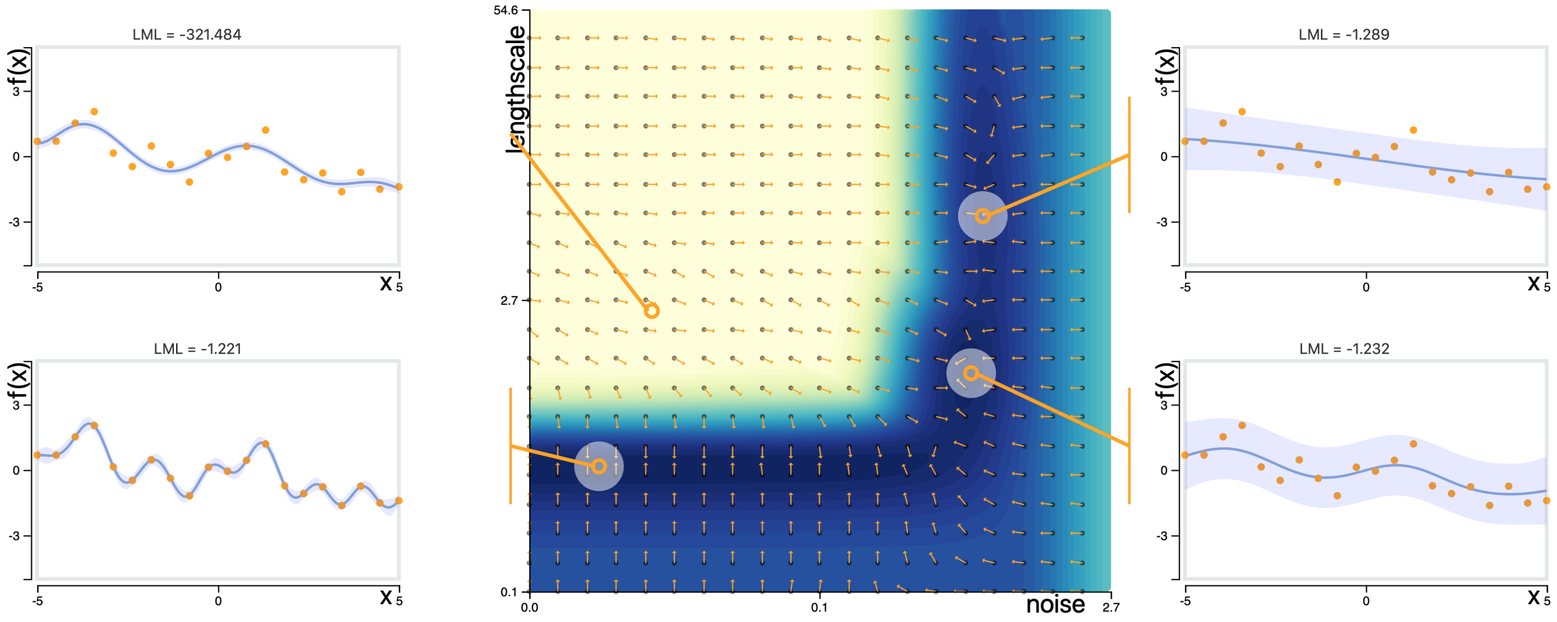
▶▶ Automatic trade-off between data fit and model complexity

Marginal Likelihood Surface



- Several plausible hyper-parameters (local optima)
- What do you expect to happen in each local optimum?

Marginal Likelihood Surface



- Several plausible hyper-parameters (local optima)
- What do you expect to happen in each local optimum?

Demo

https://drafts.distill.pub/gp/

Marginal Likelihood and Parameter Learning

- The marginal likelihood is non-convex

Marginal Likelihood and Parameter Learning

- The marginal likelihood is non-convex
- Especially in the very-small-data regime, a GP can end up in three different situations when optimizing the hyper-parameters:

Marginal Likelihood and Parameter Learning

- The marginal likelihood is non-convex
- Especially in the very-small-data regime, a GP can end up in three different situations when optimizing the hyper-parameters:
  - Short length-scales, low noise (highly nonlinear mean function with little noise)

Marginal Likelihood and Parameter Learning

- The marginal likelihood is non-convex
- Especially in the very-small-data regime, a GP can end up in three different situations when optimizing the hyper-parameters:
    - Short length-scales, low noise (highly nonlinear mean function with little noise)
    - Long length-scales, high noise (everything is considered noise)

Marginal Likelihood and Parameter Learning

- The marginal likelihood is non-convex
- Especially in the very-small-data regime, a GP can end up in three different situations when optimizing the hyper-parameters:
  - Short length-scales, low noise (highly nonlinear mean function with little noise)
  - Long length-scales, high noise (everything is considered noise)
  - Hybrid

Marginal Likelihood and Parameter Learning

- The marginal likelihood is non-convex
- Especially in the very-small-data regime, a GP can end up in three different situations when optimizing the hyper-parameters:
  - Short length-scales, low noise (highly nonlinear mean function with little noise)
  - Long length-scales, high noise (everything is considered noise)
  - Hybrid
- Re-start hyper-parameter optimization from random initialization to mitigate the problem

Marginal Likelihood and Parameter Learning

- The marginal likelihood is non-convex
- Especially in the very-small-data regime, a GP can end up in three different situations when optimizing the hyper-parameters:
  - Short length-scales, low noise (highly nonlinear mean function with little noise)
  - Long length-scales, high noise (everything is considered noise)
  - Hybrid
- Re-start hyper-parameter optimization from random initialization to mitigate the problem
- With increasing data set size the GP typically ends up in the "hybrid" mode. Other modes are unlikely.

Marginal Likelihood and Parameter Learning

- The marginal likelihood is non-convex
- Especially in the very-small-data regime, a GP can end up in three different situations when optimizing the hyper-parameters:
  - Short length-scales, low noise (highly nonlinear mean function with little noise)
  - Long length-scales, high noise (everything is considered noise)
  - Hybrid
- Re-start hyper-parameter optimization from random initialization to mitigate the problem
- With increasing data set size the GP typically ends up in the "hybrid" mode. Other modes are unlikely.
- Ideally, we would integrate the hyper-parameters out

  No closed-form solution ▶▶ Markov chain Monte Carlo

Why Does the Marginal Likelihood Work?

- Overall goal: Good generalization performance on unseen test data

Why Does the Marginal Likelihood Work?

- Overall goal: Good generalization performance on unseen test data
- Minimizing training error is not a good idea (e.g., maximum likelihood) ⏩ Overfitting
- Just adding uncertainty does not help either if the model is wrong, but it makes predictions more cautious

Why Does the Marginal Likelihood Work?

- Overall goal: Good generalization performance on unseen test data
- Minimizing training error is not a good idea (e.g., maximum likelihood) ⏩ Overfitting
- Just adding uncertainty does not help either if the model is wrong, but it makes predictions more cautious
- Marginal likelihood seems to find a good balance between fitting the data and finding a simple model (Occam's razor)

Why does the marginal likelihood lead to models that generalize well?

Marginal Likelihood: Incremental Prediction

$$\underline{p(a,b) = p(a|b)\,p(b)}$$

- "Probability of the training data" given the parameters
- General factorization (ignoring inputs $\boldsymbol{X}$):

marginal likelihood

$$p(\underline{\boldsymbol{y}}|\boldsymbol{\theta}) = p(\underline{y_1, \ldots, y_N}|\boldsymbol{\theta}) = \underbrace{\underbrace{p(y_1|\theta)\,p(y_2|y_1,\theta)}_{p(y_1,y_2|\theta)}\,p(y_3|y_1,y_2,\theta)}_{p(y_1,y_2,y_3|\theta)}$$

$$\cdots\cdots\; p(y_N|y_1,\ldots,y_{N-1},\theta)$$

Marginal Likelihood: Incremental Prediction

- "Probability of the training data" given the parameters
- General factorization (ignoring inputs $\boldsymbol{X}$):

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{\theta}) &= p(y_1, \dots, y_N|\boldsymbol{\theta}) \\ &= p(y_1|\boldsymbol{\theta})p(y_2|y_1, \boldsymbol{\theta})p(y_3|y_1, y_2, \boldsymbol{\theta}) \cdot \dots \cdot p(y_N|y_1, \dots, y_{N-1}, \boldsymbol{\theta}) \end{aligned}$$

Marginal Likelihood: Incremental Prediction

- "Probability of the training data" given the parameters
- General factorization (ignoring inputs $\boldsymbol{X}$):

$$
\begin{aligned}
p(\boldsymbol{y}|\boldsymbol{\theta}) &= p(y_1, \dots, y_N|\boldsymbol{\theta}) \\
&= p(y_1|\boldsymbol{\theta})p(y_2|y_1, \boldsymbol{\theta})p(y_3|y_1, y_2, \boldsymbol{\theta}) \cdot \ldots \cdot p(y_N|y_1, \dots, y_{N-1}, \boldsymbol{\theta}) \\
&= p(y_1|\boldsymbol{\theta}) \prod_{n=2}^{N} p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})
\end{aligned}
$$

Marginal Likelihood: Incremental Prediction (2)

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^{N} p(y_n|y_1, \ldots, y_{n-1}, \boldsymbol{\theta})$$

Marginal Likelihood: Incremental Prediction (2)

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^{N} p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})$$

- If we think of this as a sequence model (where data arrives sequentially), the marginal likelihood predicts the nth training observation given all “previous” observations

Marginal Likelihood: Incremental Prediction (2)

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^{N} p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})$$

- If we think of this as a sequence model (where data arrives sequentially), the marginal likelihood predicts the nth training observation given all "previous" observations
- Predict training data y_n that has not been accounted for (we only condition on $y_1, \dots, y_{n-1}$) ⏩ Treat next data point as test data

Marginal Likelihood: Incremental Prediction (2)

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^{N} p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})$$

- If we think of this as a sequence model (where data arrives sequentially), the marginal likelihood predicts the nth training observation given all "previous" observations
- Predict training data y_n that has not been accounted for (we only condition on $y_1, \dots, y_{n-1}$) ⏩ Treat next data point as test data
- Intuition: If it continuously predicted well on all N previous points, it probably will do well next time

Marginal Likelihood: Incremental Prediction (2)

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^{N} p(y_n|y_1, \ldots, y_{n-1}, \boldsymbol{\theta})$$

- If we think of this as a sequence model (where data arrives sequentially), the marginal likelihood predicts the nth training observation given all “previous” observations
- Predict training data y_n that has not been accounted for (we only condition on $y_1, \ldots, y_{n-1}$) ⏩ Treat next data point as test data
- Intuition: If it continuously predicted well on all N previous points, it probably will do well next time
  ⏩ Proxy for generalization error on unseen test data

Marginal Likelihood: Illustration

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = p(y_1, \dots, y_N|\boldsymbol{\theta}) = p(y_1|\boldsymbol{\theta}) \prod_{n=2}^{N} p(y_n|y_1, \dots, y_{n-1}, \boldsymbol{\theta})$$

Marginal Likelihood Computation in Action[2]

- Short length-scale

[2] Thanks to Mark van der Wilk

Marginal Likelihood Computation in Action[3]

- Long length-scale

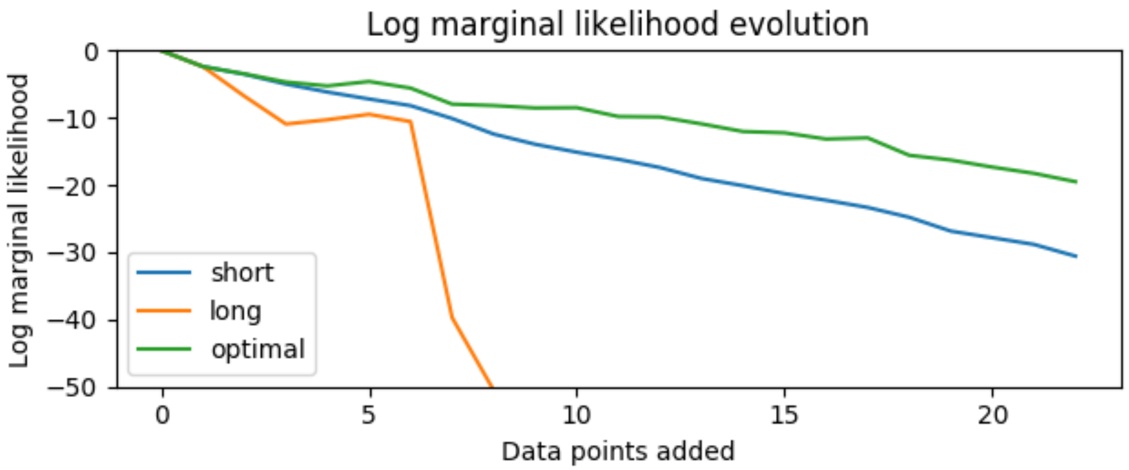
[3] Thanks to Mark van der Wilk

Marginal Likelihood Computation in Action[4]

- Optimal length-scale

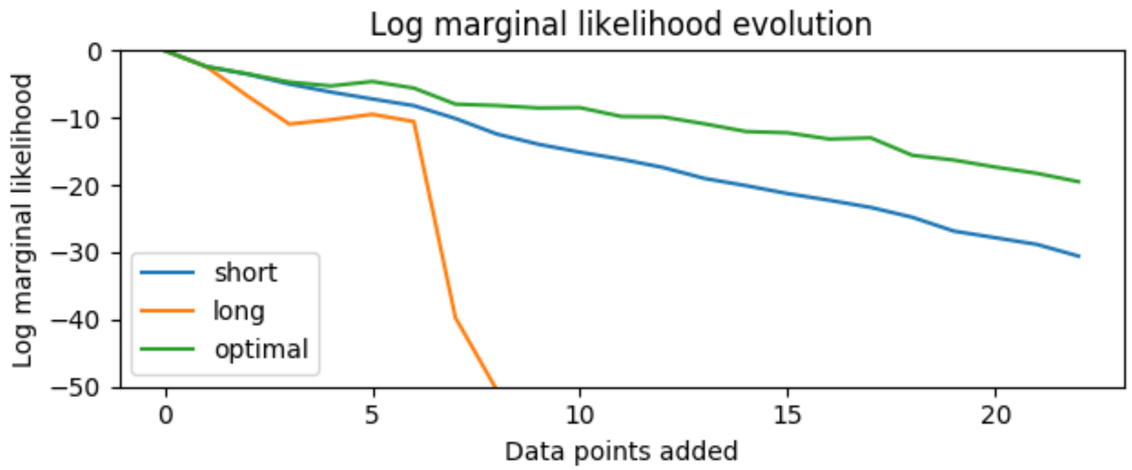
[4] Thanks to Mark van der Wilk

Marginal Likelihood Evolution



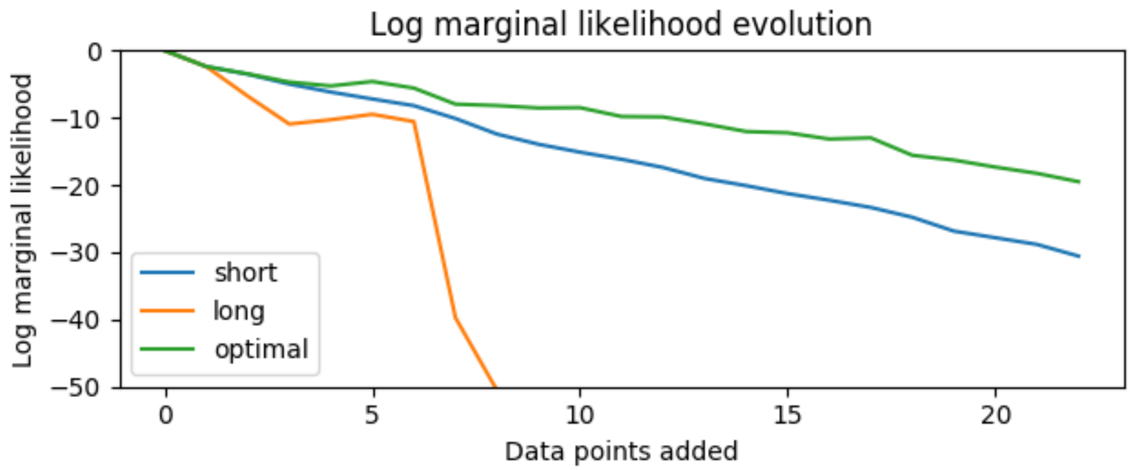
- Short lengthscale: consistently overestimates variance
  - No high density, even with observations inside the error bars

Marginal Likelihood Evolution



- Short lengthscale: consistently overestimates variance
  - No high density, even with observations inside the error bars
- Long lengthscale: consistently underestimates variance
  - Low density because observations are outside the error bars

Marginal Likelihood Evolution



- Short lengthscale: consistently overestimates variance
  - No high density, even with observations inside the error bars
- Long lengthscale: consistently underestimates variance
  - Low density because observations are outside the error bars
- Optimal lengthscale: trades off both behaviors reasonably well

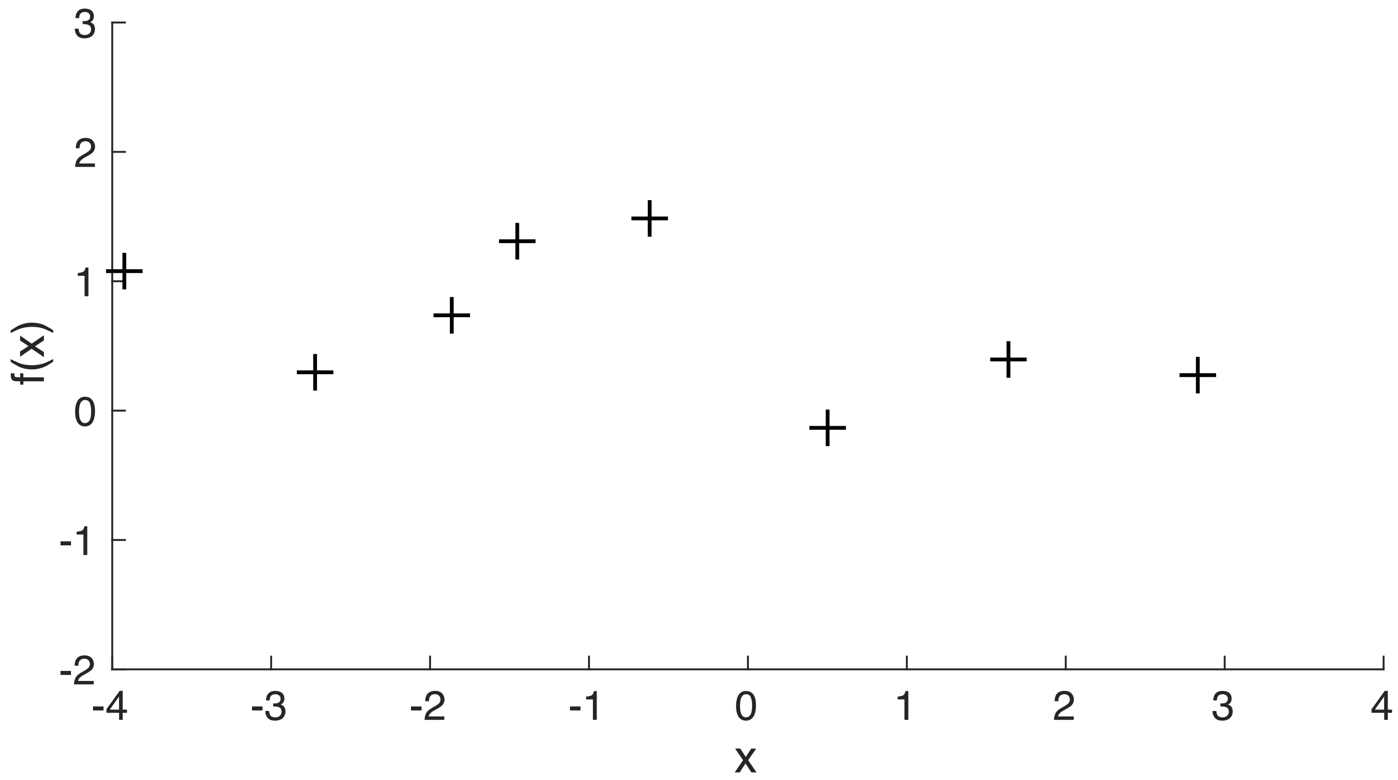
Model Selection—Mean Function and Kernel

- Assume we have a finite set of models M_i, each one specifying a mean function m_i and a kernel k_i. How do we find the best one?

Model Selection—Mean Function and Kernel

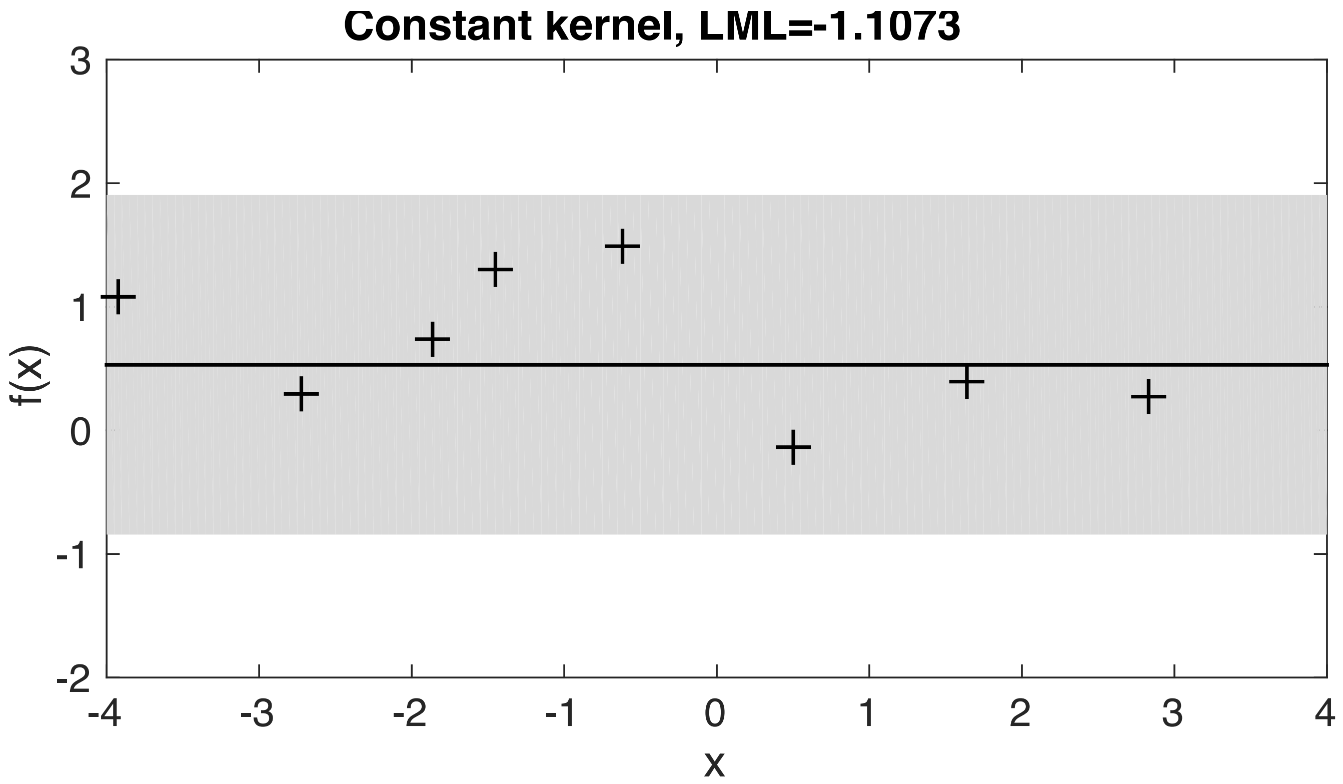
- Assume we have a finite set of models M_i, each one specifying a mean function m_i and a kernel k_i. How do we find the best one?
- Some options:
  - Cross validation
  - Bayesian Information Criterion, Akaike Information Criterion
  - Compare marginal likelihood values (assuming a uniform prior on the set of models)

Example



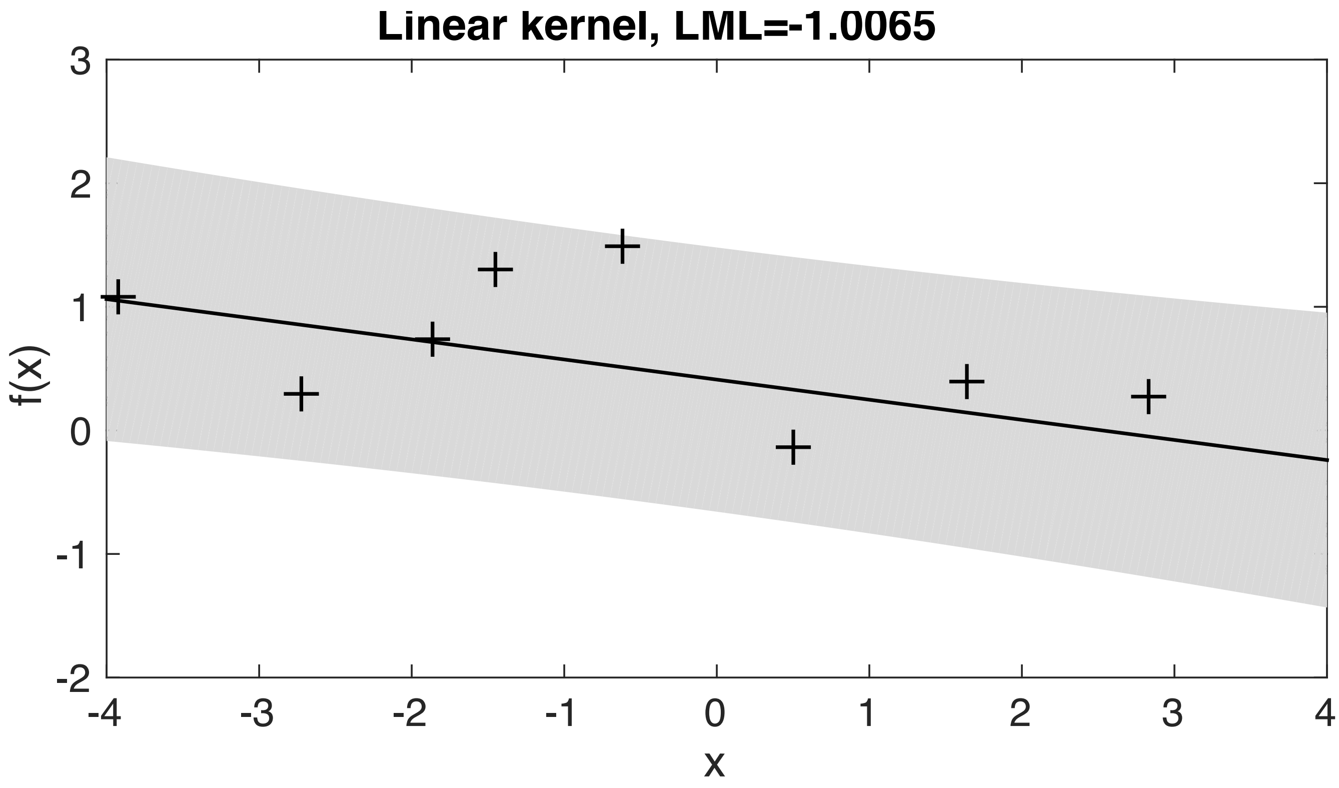
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

Example



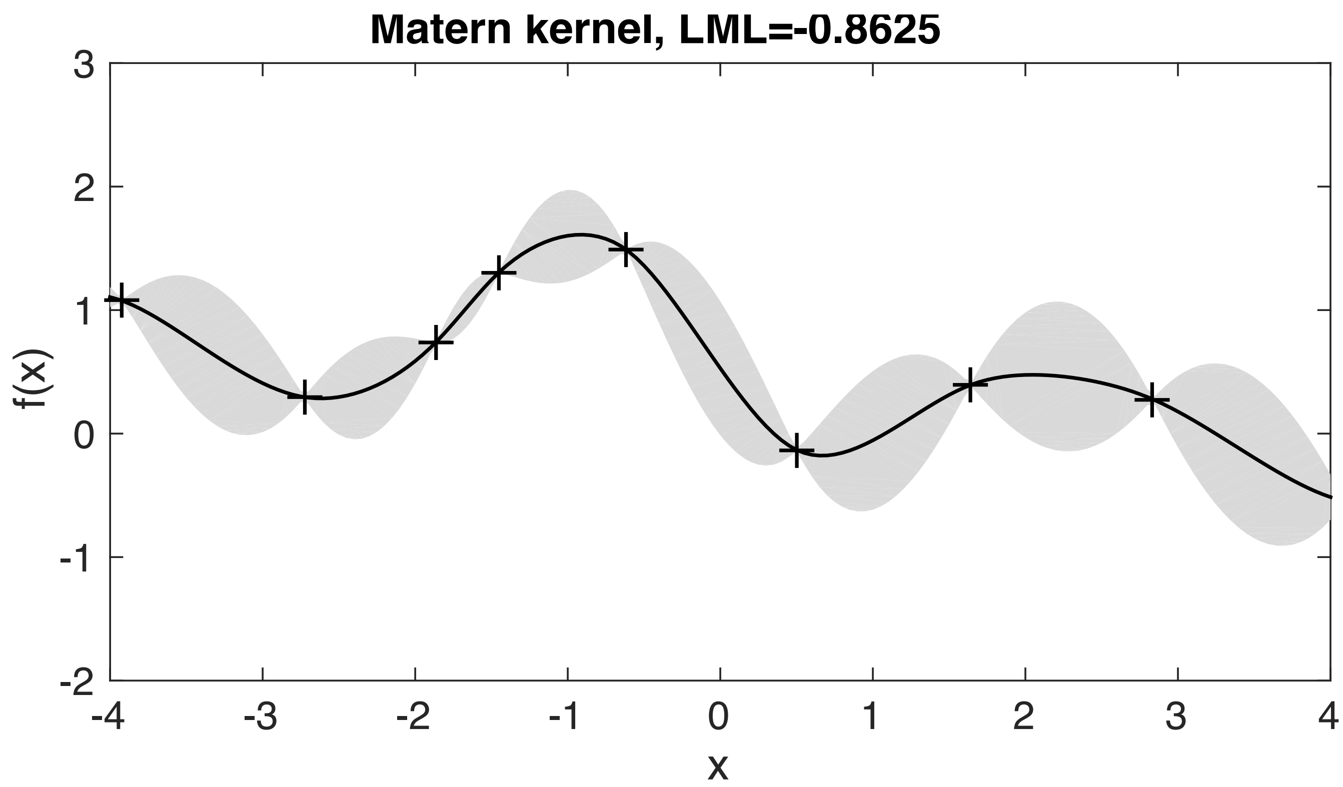
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

Example



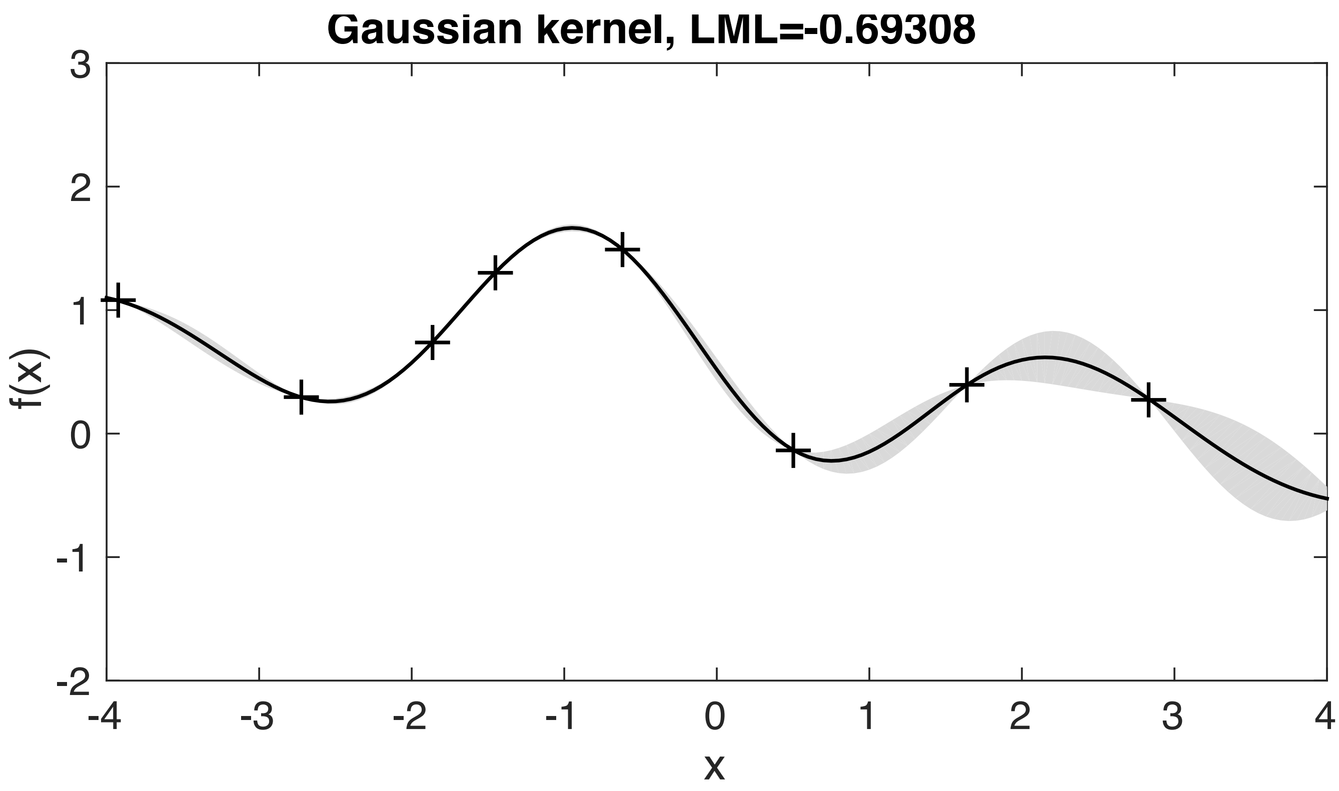
- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

Example



- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

Example



- Four different kernels (mean function fixed to $m \equiv 0$)
- MAP hyper-parameters for each kernel
- Log-marginal likelihood values for each (optimized) model

GP Training: Illustration[5]

- Prior: $f(\boldsymbol{x}) = \theta_s f_{\text{smooth}}(\boldsymbol{x}) + \theta_p f_{\text{periodic}}(\boldsymbol{x})$, with smooth and periodic GP priors, respectively.

[5]Thanks to Mark van der Wilk

GP Training: Illustration[5]

- Prior: $f(\boldsymbol{x}) = \theta_s f_{\text{smooth}}(\boldsymbol{x}) + \theta_p f_{\text{periodic}}(\boldsymbol{x})$, with smooth and periodic GP priors, respectively.
- Amount of periodicity vs. smoothness is automatically chosen by selecting hyper-parameters θ_s, θ_p.
- Marginal likelihood learns how to generalize, not just to fit the data

[5] Thanks to Mark van der Wilk

Limitations and Guidelines

Limitations of Gaussian Processes

Computational and memory complexity

Training set size: N

- Training scales in $\mathcal{O}(N^3)$
- Prediction (variances) scales in $\mathcal{O}(N^2)$
- Memory requirement: $\mathcal{O}(ND + N^2)$

▶▶ **Practical limit** $N \approx 10,000$

Limitations of Gaussian Processes

Computational and memory complexity

Training set size: N

- Training scales in $\mathcal{O}(N^3)$
- Prediction (variances) scales in $\mathcal{O}(N^2)$
- Memory requirement: $\mathcal{O}(ND + N^2)$

▶▶ **Practical limit** $N \approx 10,000$

Some solution approaches:

- Sparse GPs with inducing variables (e.g., Snelson & Ghahramani, 2006; Quiñonero-Candela & Rasmussen, 2005; Titsias 2009; Hensman et al., 2013; Matthews et al., 2016)
- Combination of local GP expert models (e.g., Tresp 2000; Cao & Fleet 2014; Deisenroth & Ng, 2015)
- Variational Fourier features (Hensman et al., 2018)

Tips and Tricks for Practitioners

- To set initial hyper-parameters, use domain knowledge.

⏩ `https://drafts.distill.pub/gp`

Tips and Tricks for Practitioners

- To set initial hyper-parameters, use domain knowledge.
- Standardize input data and set initial length-scales ℓ to ≈ 0.5.

⏩ `https://drafts.distill.pub/gp`

Tips and Tricks for Practitioners

- To set initial hyper-parameters, use domain knowledge.
- Standardize input data and set initial length-scales ℓ to ≈ 0.5.
- Standardize targets y and set initial signal variance to $\sigma_f \approx 1$.

⏩ `https://drafts.distill.pub/gp`

Tips and Tricks for Practitioners

- To set initial hyper-parameters, use domain knowledge.
- Standardize input data and set initial length-scales ℓ to ≈ 0.5.
- Standardize targets y and set initial signal variance to $\sigma_f \approx 1$.
- Often useful: Set initial noise level relatively high (e.g., $\sigma_n \approx 0.5 \times \sigma_f$ amplitude), even if you think your data have low noise. The optimization surface for your other parameters will be easier to move in.

⏩ `https://drafts.distill.pub/gp`

Tips and Tricks for Practitioners

- To set initial hyper-parameters, use domain knowledge.
- Standardize input data and set initial length-scales ℓ to ≈ 0.5.
- Standardize targets y and set initial signal variance to $\sigma_f \approx 1$.
- Often useful: Set initial noise level relatively high (e.g., $\sigma_n \approx 0.5 \times \sigma_f$ amplitude), even if you think your data have low noise. The optimization surface for your other parameters will be easier to move in.
- When optimizing hyper-parameters, try random restarts or other tricks to avoid local optima are advised.

▶▶ `https://drafts.distill.pub/gp`

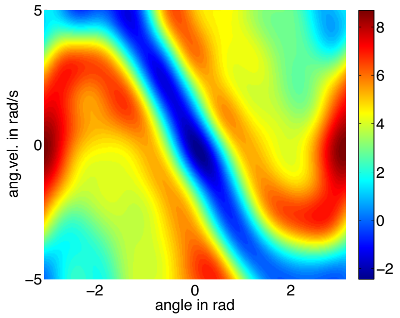
Tips and Tricks for Practitioners

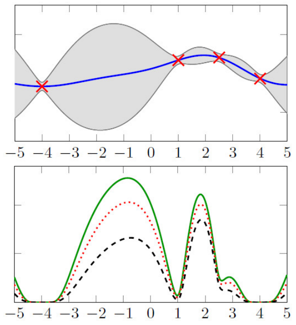
- To set initial hyper-parameters, use domain knowledge.
- Standardize input data and set initial length-scales ℓ to ≈ 0.5.
- Standardize targets y and set initial signal variance to $\sigma_f \approx 1$.
- Often useful: Set initial noise level relatively high (e.g., $\sigma_n \approx 0.5 \times \sigma_f$ amplitude), even if you think your data have low noise. The optimization surface for your other parameters will be easier to move in.
- When optimizing hyper-parameters, try random restarts or other tricks to avoid local optima are advised.
- Mitigate the problem of numerical instability (Cholesky decomposition of $\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$) by penalizing high signal-to-noise ratios σ_f / σ_n

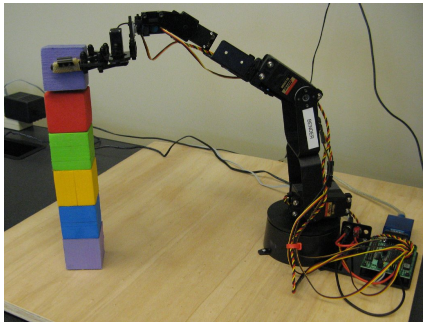
▸▸ https://drafts.distill.pub/gp

Application Areas

Application Areas

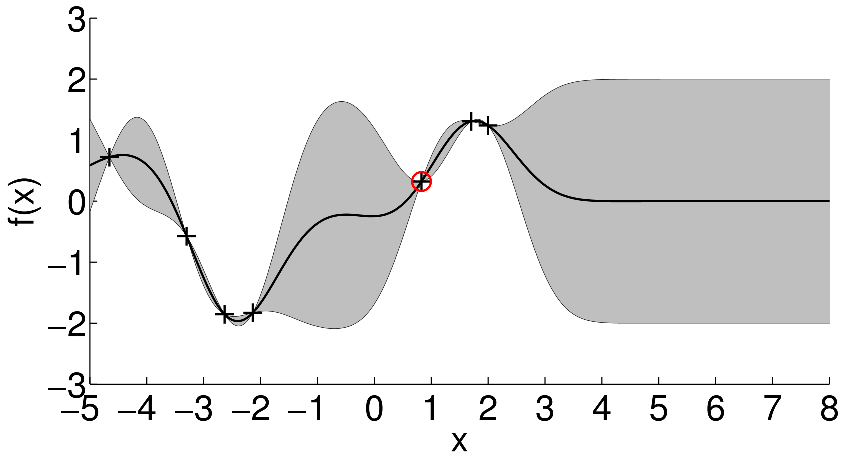


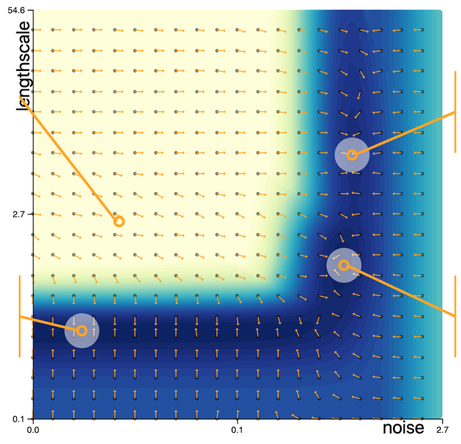




- Reinforcement learning and robotics
  - Model value functions and/or dynamics with GPs
- Bayesian optimization (Experimental Design)
  - Model unknown utility functions with GPs
- Geostatistics
  - Spatial modeling (e.g., landscapes, resources)
- Sensor networks
- Time-series modeling and forecasting

Summary





- Gaussian processes are the gold-standard for regression
- Closely related to Bayesian linear regression
- Computations boil down to manipulating multivariate Gaussian distributions
- Marginal likelihood objective automatically trades off data fit and model complexity

References I

[1] G. Bertone, M. P. Deisenroth, J. S. Kim, S. Liem, R. R. de Austri, and M. Welling. Accelerating the BSM Interpretation of LHC Data with Machine Learning. arXiv preprint arXiv:1611.02704, 2016.

[2] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold Gaussian Processes for Regression. In *Proceedings of the International Joint Conference on Neural Networks*, 2016.

[3] Y. Cao and D. J. Fleet. Generalized Product of Experts for Automatic and Principled Fusion of Gaussian Process Predictions. `http://arxiv.org/abs/1410.7827`, 2014.

[4] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley-Interscience, 1993.

[5] M. Cutler and J. P. How. Efficient Reinforcement Learning for Robots using Informative Simulated Priors. In *Proceedings of the International Conference on Robotics and Automation*, 2015.

[6] M. P. Deisenroth and J. W. Ng. Distributed Gaussian Processes. In *Proceedings of the International Conference on Machine Learning*, 2015.

[7] M. P. Deisenroth, C. E. Rasmussen, and D. Fox. Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, 2011.

[8] M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian Process Dynamic Programming. *Neurocomputing*, 72(7–9):1508–1524, Mar. 2009.

[9] M. P. Deisenroth, R. Turner, M. Huber, U. D. Hanebeck, and C. E. Rasmussen. Robust Filtering and Smoothing with Gaussian Processes. *IEEE Transactions on Automatic Control*, 57(7):1865–1871, 2012.

[10] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen. Bayesian Inference and Learning in Gaussian Process State-Space Models with Particle MCMC. In *Advances in Neural Information Processing Systems*. 2013.

[11] N. HajiGhassemi and M. P. Deisenroth. Approximate Inference for Long-Term Forecasting with Periodic Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2014.

[12] J. Hensman, N. Durrande, and A. Solin. Variational Fourier Features for Gaussian Processes. *Journal of Machine Learning Research*, pages 1–52, 2018.

References II

[13] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian Processes for Big Data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2013.

[14] A. Krause, A. Singh, and C. Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research*, 9:235–284, Feb. 2008.

[15] M. C. H. Lee, H. Salimbeni, M. P. Deisenroth, and B. Glocker. Patch Kernels for Gaussian Processes in High-Dimensional Imaging Problems. In *NIPS Workshop on Practical Bayesian Nonparametrics*, 2016.

[16] J. R. Lloyd, D. Duvenaud, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Automatic Construction and Natural-Language Description of Nonparametric Regression Models. In *AAAI Conference on Artificial Intelligence*, pages 1–11, 2014.

[17] D. J. C. MacKay. Introduction to Gaussian Processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, volume 168, pages 133–165. Springer, 1998.

[18] A. G. d. G. Matthews, J. Hensman, R. Turner, and Z. Ghahramani. On Sparse Variational Methods and the Kullback-Leibler Divergence between Stochastic Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2016.

[19] M. A. Osborne, S. J. Roberts, A. Rogers, S. D. Ramchurn, and N. R. Jennings. Towards Real-Time Information Processing of Sensor Network Data Using Computationally Efficient Multi-output Gaussian Processes. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 109–120. IEEE Computer Society, 2008.

[20] J. Quiñonero-Candela and C. E. Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6(2):1939–1960, 2005.

[21] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[22] S. Roberts, M. A. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian Processes for Time Series Modelling. *Philosophical Transactions of the Royal Society (Part A)*, 371(1984), Feb. 2013.

References III

[23] B. Schölkopf and A. J. Smola. *Learning with Kernels—Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA, 2002.

[24] E. Snelson and Z. Ghahramani. Sparse Gaussian Processes using Pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1257–1264. The MIT Press, Cambridge, MA, USA, 2006.

[25] M. K. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.

[26] V. Tresp. A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741, 2000.

[27] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep Kernel Learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2016.