AutoML: Gaussian Processes

Gaussian Process Training

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Training of a Gaussian Process

- To make predictions for a regression task by a Gaussian process, one needs to perform matrix computations.
- Here, the main difficulty is how to do **model selection**, i.e., how to choose the best covariance function and how to tune the hyperparameters.
- There is a multitude of possible families of covariance functions, each coming with a number of hyperparameters to be chosen.
- We will refer to the model selection as **training** of a Gaussian process.

Training a GP via the Maximum Likelihood I

- Let us assume $y = f(\mathbf{x}) + \epsilon,\ \epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$, where $f(\mathbf{x}) \sim \mathcal{G}\left(\mathbf{0}, k\left(\mathbf{x}, \mathbf{x}' \mid \boldsymbol{\theta}\right)\right)$.

- Noticing that $\boldsymbol{y} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{K} + \sigma^2 \boldsymbol{I}\right)$, we can find the marginal log-likelihood (or evidence):

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

with $\boldsymbol{K}_y := \boldsymbol{K} + \sigma^2 \boldsymbol{I}$ and $\boldsymbol{\theta}$ denoting the parameters of the covariance function (i.e., the hyperparameters).

Training a GP via the Maximum Likelihood II

Recalling that the increase of the length-scale reduces the model flexibility, the three terms of the marginal likelihood can be interpreted as follows.

- The data fit $-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}_y^{-1}\boldsymbol{y}$. The data fit tends to decrease by increasing the length-scale.
- The complexity penalty $-\frac{1}{2}\,log\,|\boldsymbol{K}_y|$, which depends on the covariance function. This term also decreases with the increase of the length-scale (the model gets less complex as the length-scale grows).
- The normalization constant $-\frac{n}{2}\,log\,2\pi$.

Training a GP: Example I

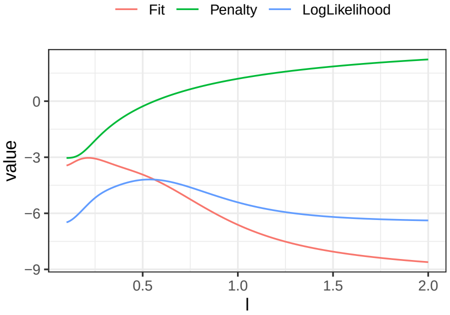
To visualize the said ideas, let us consider a zero-mean Gaussian process with a squared exponential kernel:

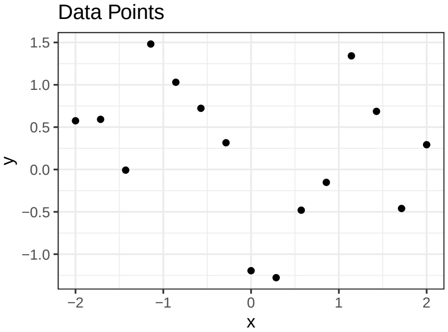
$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2\ell^2}\|\mathbf{x} - \mathbf{x}'\|^2\right).$$

- Recall that the model becomes smoother and less complex as the length-scale ℓ increases.

- We will show how each of the following terms behaves if the value of ℓ increases:
  - the data fit $-\frac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}_y^{-1}\boldsymbol{y}$,
  - the complexity penalty $-\frac{1}{2}\, log\, |\boldsymbol{K}_y|$,
  - the overall value of the marginal likelihood $log\, p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$.

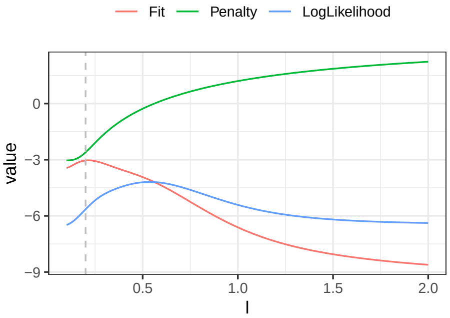
Training a GP: Example II

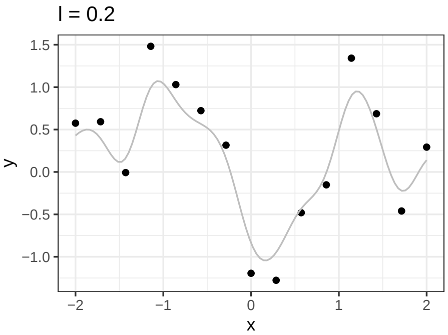




ℹ The left plot depicts how the data fit, the complexity penalty (a higher value means less penalization), and the overall marginal likelihood behave for increasing values of the length-scale.

Training a GP: Example III

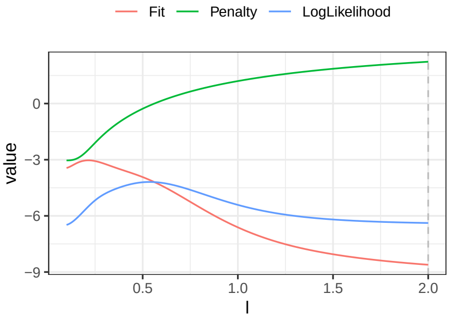


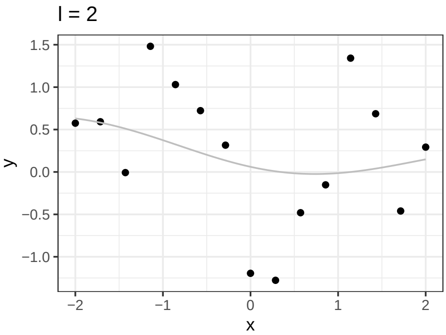


ℹ The left plot depicts how the data fit, the complexity penalty (a higher value means less penalization), and the overall marginal likelihood behave for increasing values of the length-scale.

A small ℓ leads to a good fit, but, to a high complexity penalty.

Training a GP: Example IV

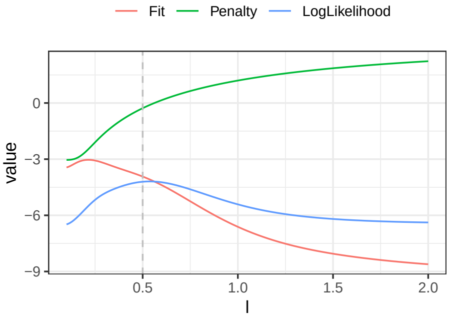


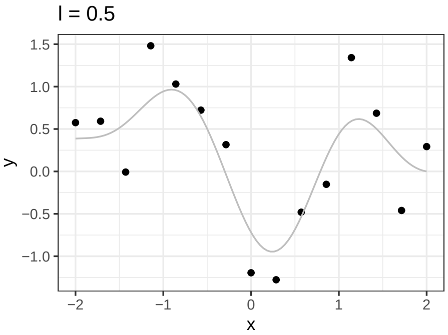


ℹ The left plot depicts how the data fit, the complexity penalty (a higher value means less penalization), and the overall marginal likelihood behave for increasing values of the length-scale.

💡 A large ℓ results in a poor fit.

Training a GP: Example V





The left plot depicts how the data fit, the complexity penalty (a higher value means less penalization), and the overall marginal likelihood behave for increasing values of the length-scale.

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

Training a GP via the Maximum Likelihood I

To choose the hyperparameters by maximizing the marginal likelihood, we need to find the partial derivatives of the likelihood w.r.t. the hyperparameters:

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} \, log \, p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) \quad &= \quad \frac{\partial}{\partial \theta_j} \left(-\frac{1}{2} \boldsymbol{y}^\top \boldsymbol{K}_y^{-1} \boldsymbol{y} - \frac{1}{2} \, log \, |\boldsymbol{K}_y| - \frac{n}{2} \, log \, 2\pi \right) \\
&= \quad \frac{1}{2} \boldsymbol{y}^\top \boldsymbol{K}^{-1} \frac{\partial \boldsymbol{K}}{\partial \theta_j} \boldsymbol{K}^{-1} \boldsymbol{y} - \frac{1}{2} \text{tr} \left(\boldsymbol{K}^{-1} \frac{\partial \boldsymbol{K}}{\partial \boldsymbol{\theta}} \right) \\
&= \quad \frac{1}{2} \text{tr} \left((\boldsymbol{K}^{-1} \boldsymbol{y} \boldsymbol{y}^\top \boldsymbol{K}^{-1} - \boldsymbol{K}^{-1}) \frac{\partial \boldsymbol{K}}{\partial \theta_j} \right)
\end{aligned}
$$

Above, we used the following identities:

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

Training a GP via the Maximum Likelihood II

- The complexity and the runtime of training a Gaussian process is dominated by the computational task of inverting $\boldsymbol{K}$.

- Standard methods require $\mathcal{O}(n^3)$ time (!) for inverting an $n \times n$ matrix.

- Once $\boldsymbol{K}^{-1}$ 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, and using a gradient based optimizer is advantageous.

Training a GP via the Maximum Likelihood III

Workarounds to make GP estimation feasible for big data include:

- Using kernels that yield sparse $\boldsymbol{K}$: 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**; $\mathcal{O}(nm^2)$.
- Exploiting low-rank approximations of $\boldsymbol{K}$ by using only a representative subset (enducing points) of m training data $\boldsymbol{X}_m$:**Nyström approximation** $\boldsymbol{K} \approx \boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-}\boldsymbol{K}_{mn}$, with $\mathcal{O}(nmk + m^3)$ for a rank-k-approximate inverse of $\boldsymbol{K}_{mm}$.
- Utilizing structure in $\boldsymbol{K}$ induced by the kernel: exact solutions but complicated maths, not applicable for all kernels.

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