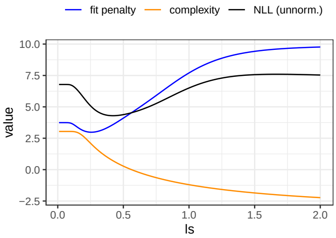
Introduction to Machine Learning

Gaussian Processes
Training of a Gaussian Process



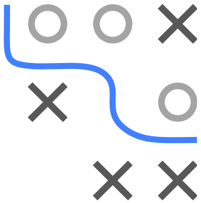
Learning goals

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

TRAINING OF A GAUSSIAN PROCESS

- All we need for GP predictions (in regression): matrix computations
- Implicit assumption: fully specified cov function, incl. hyperparams
- Nice GP property: numerical hyperparams of given cov function can be learned during training

TRAINING VIA MARGINAL LIKELIHOOD



- Let
$$y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$
with $f \sim \mathcal{GP}(\mathbf{0}, k(\cdot, \cdot|\boldsymbol{\theta}))$ for hyperparam config $\boldsymbol{\theta}$

- This yields $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_y)$ with $\mathbf{K}_y = \mathbf{K} + \sigma^2 \boldsymbol{I}$

- We get the **negative marginal log-likelihood / evidence**
$$\begin{aligned} -\log p(\mathbf{y}|\boldsymbol{X}, \boldsymbol{\theta}) &= -\log \left[(2\pi)^{-n/2} |\mathbf{K}_y|^{-1/2} \exp(-\tfrac{1}{2}\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{y})\right] \\ &= \tfrac{1}{2}\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{y} + \tfrac{1}{2}\log|\mathbf{K}_y| + \tfrac{n}{2}\log 2\pi \end{aligned}$$

- $-\log p(\mathbf{y}|\boldsymbol{X}, \boldsymbol{\theta})$ depends on $\boldsymbol{\theta}$ via $\mathbf{K}_y \Rightarrow$ optimize for $\boldsymbol{\theta}$

- GP training optimizes kernel hyperparams by minimizing the neg. marginal likelihood, balancing data fit ($\tfrac{1}{2}\mathbf{y}^\top\mathbf{K}_y^{-1}\mathbf{y}$) and model complexity ($\tfrac{1}{2}\log|\mathbf{K}_y|$)

NEGATIVE LOG-LIKELIHOOD COMPONENTS

- Consider common cov type parameterized by length-scale: $\boldsymbol{\theta} = \ell$ (for simplicity: assume univariate rather than different for each dim)

- E.g., squared exponential kernel

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp(-\frac{1}{2\ell^2}\|\mathbf{x} - \tilde{\mathbf{x}}\|^2)$$

- For small ℓ, cov decays quickly $\Rightarrow$ local model, $\mathbf{K}_y$ approaches $\sigma^2 \boldsymbol{I}$

- **Data fit**: small $\ell \Rightarrow$ small $\mathbf{K}_y^{-1}\mathbf{y}$ (cov structure aligns well with observed data; small residuals) $\Rightarrow$ small $\frac{1}{2}\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{y}$

- **Complexity penalty**: small $\ell \Rightarrow$ large $\frac{1}{2}\log|\mathbf{K}_y|$

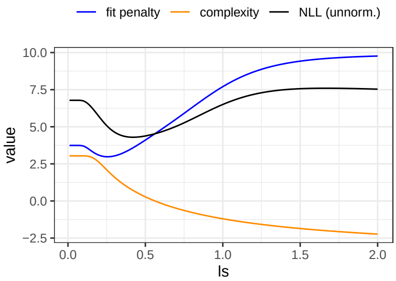
- Normalization constant $\frac{n}{2}\log 2\pi$

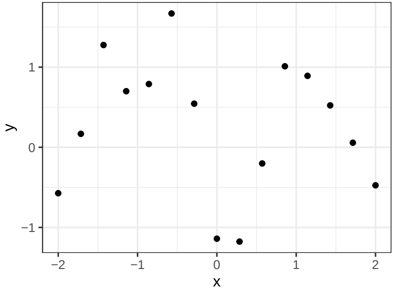
NLL COMPONENTS: EXAMPLE

- Let $f \sim \mathcal{GP}(\mathbf{0}, k(\cdot, \cdot))$ with squared exp kernel

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp(-\frac{1}{2\ell^2}\|\mathbf{x} - \tilde{\mathbf{x}}\|^2)$$

- Minimize NLL by trading off data fit & complexity



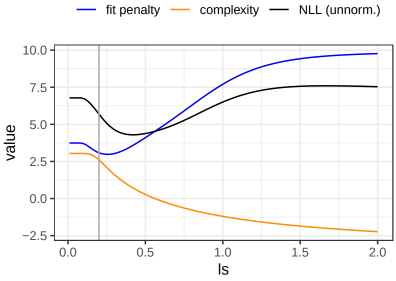


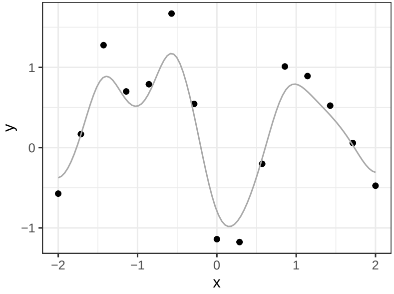
NLL COMPONENTS: EXAMPLE

- Let $f \sim \mathcal{GP}(\mathbf{0}, k(\cdot,\cdot))$ with squared exp kernel

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp(-\frac{1}{2\ell^2}\|\mathbf{x} - \tilde{\mathbf{x}}\|^2)$$

- Minimize NLL by trading off data fit & complexity
- $\ell = 0.2$: good fit but high complexity





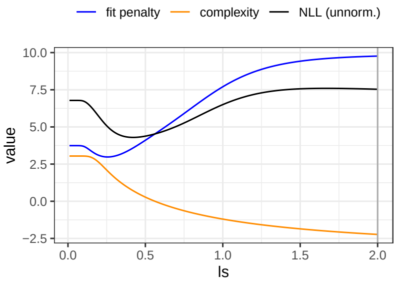
NLL COMPONENTS: EXAMPLE

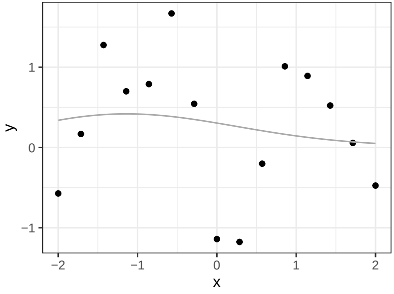
- Let $f \sim \mathcal{GP}(\mathbf{0}, k(\cdot,\cdot))$ with squared exp kernel

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp(-\frac{1}{2\ell^2}\|\mathbf{x} - \tilde{\mathbf{x}}\|^2)$$

- Minimize NLL by trading off data fit & complexity

- $\ell = 2$: smooth but poor fit





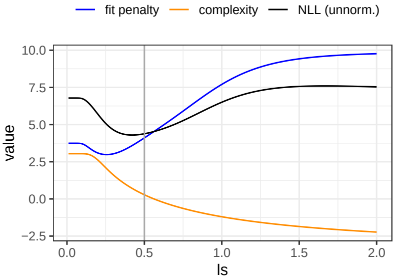
NLL COMPONENTS: EXAMPLE

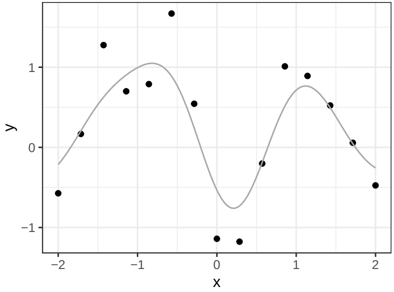
- Let $f \sim \mathcal{GP}(\mathbf{0}, k(\cdot,\cdot))$ with squared exp kernel

$$k(\mathbf{x}, \tilde{\mathbf{x}}) = \exp(-\frac{1}{2\ell^2}\|\mathbf{x} - \tilde{\mathbf{x}}\|^2)$$

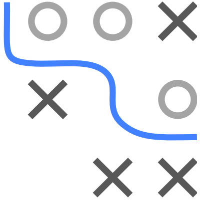
- Minimize NLL by trading off data fit & complexity

- $\ell = 0.5$: balancing data fit and smoothness





OPTIMIZING KERNEL HYPERPARAMETERS



- Set partial derivatives wrt hyperparams to 0

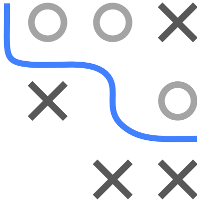
$$\begin{aligned}\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}\mid \boldsymbol{X},\boldsymbol{\theta}) &= \frac{\partial}{\partial\theta_j}\left(-\tfrac{1}{2}\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{y}-\tfrac{1}{2}\log|\mathbf{K}_y|-\tfrac{n}{2}\log 2\pi\right)\\ &= \tfrac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\theta_j}\mathbf{K}^{-1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

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

- Bottleneck: inverting (or rather, decomposing) $\mathbf{K}$
  $\Rightarrow \mathcal{O}(n^3)$ for standard methods

- Only $\mathcal{O}(n^2)$ per hyperparam / partial derivative once $\mathbf{K}^{-1}$ is known
  $\Rightarrow$ small overhead, so use gradient-based optim

STRATEGIES FOR BIG DATA



- Kernels that yield sparse $\mathbf{K} \Rightarrow$ cheaper to invert
- Subsample data $\Rightarrow \mathcal{O}(m^3)$ with $m^3 \ll n^3$
- **Bayesian committee**: combine estimates on different size-m estimates $\Rightarrow \mathcal{O}(nm^2)$
- **Nyström approx**: low-rank approx from representative subset ("inducing points"): $\mathbf{K} \approx \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn} \Rightarrow \mathcal{O}(nmk + m^3)$ for rank-k-approx inverse of $\mathbf{K}_{mm}$
- Exploit structure in $\mathbf{K}$ induced by kernels $\Rightarrow$ exact but complicated solutions; kernel-specific
- Still active research area