Chapter 04: Gaussian Processes
Pages: 67-77

Probabilistic Artificial Intelligence

Notes by Kumar Anurag

4.4 Model Selection

- So far, we assumed hyperparameters θ (e.g., kernel parameters) are known.
- In practice, we must learn θ from data.
- A common supervised learning technique:
  - Pick θ such that the resulting predictor $\hat{f}_\theta$ performs best on a hold-out validation set.
- This is known as **point estimate-based model selection**.
- Later, we'll contrast it with a fully **Bayesian approach** that uses the posterior over f instead of a point estimate.

4.4.1 Optimizing Validation Performance

- Split data $\mathcal{D}$ into:
  - Training set $\mathcal{D}^{\text{train}} = \{(x_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^n$
  - Validation set $\mathcal{D}^{\text{val}} = \{(x_j^{\text{val}}, y_j^{\text{val}})\}_{j=1}^m$
- Step 1: Train a model $\hat{f}_j$ for some candidate θ_j on the training set:

$$\hat{f}_j \triangleq \arg\max_f p(f \mid x_{1:n}^{\text{train}}, y_{1:n}^{\text{train}}) \tag{4.22}$$

- Step 2: Select $\hat{\theta}$ based on validation performance:

$$\hat{\theta} \triangleq \arg\max_{\theta_j} p(y_{1:m}^{\text{val}} \mid x_{1:m}^{\text{val}}, \hat{f}_j) \tag{4.23}$$

Why Validation Sets Help: Remark 4.8

Approximating Population Risk

Minimizing loss on the same data used for training often leads to overfitting.
Instead, use independent validation data to estimate risk:

$$\frac{1}{m}\sum_{i=1}^{m}\ell(y_i^{\text{val}} \mid x_i^{\text{val}}, \hat{f}_j) \approx \mathbb{E}_{(x,y)\sim\mathcal{P}}\left[\ell(y \mid x, \hat{f}_j)\right] \tag{4.24}$$

- This is an empirical approximation of the true population risk.
- Helps prevent overfitting and ensures generalization.

Limitation: Still relies on a *point estimate* $\hat{f}_j$. Can we do better?

Maximizing the Marginal Likelihood

- In Bayesian regression, we don't want to pick a single function $\hat{f}$.
- Instead, we want to score how likely the observed data is under all possible functions defined by kernel parameters θ.

Marginal likelihood:

$$p(y_{1:n} \mid x_{1:n}, \theta) = \int p(y_{1:n} \mid x_{1:n}, f, \theta) \cdot p(f \mid \theta)\, df \tag{4.26}$$

Key idea: We integrate over all functions f rather than picking just one.

This gives us a more robust measure of how "compatible" θ is with the observed data.

Why This Works — Intuition Behind the Math

Marginal likelihood prefers models that:

- Fit the data well $\rightarrow$ high likelihood
- Are not overly complex $\rightarrow$ high prior probability

	likelihood	prior
"underfit" model (too simple $\boldsymbol{\theta}$)	small for "almost all" f	large
"overfit" model (too complex $\boldsymbol{\theta}$)	large for "few" f small for "most" f	small
"just right"	moderate for "many" f	moderate

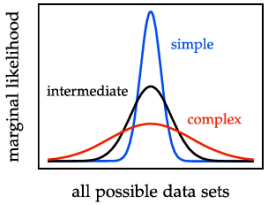


Figure 4.8: A schematic illustration of the marginal likelihood of a simple, intermediate, and complex model across all possible data sets.

Conclusion: Maximizing marginal likelihood automatically balances these two forces.

Marginal Likelihood for GPs

Recall from GP prior:

$$y_{1:n} \mid x_{1:n}, \theta \sim \mathcal{N}(0, K_{f,\theta} + \sigma_n^2 I) \tag{4.27}$$

Let $K_{y,\theta} = K_{f,\theta} + \sigma_n^2 I$

Then the marginal likelihood becomes:

$$p(y \mid x, \theta) = \mathcal{N}(y \mid 0, K_{y,\theta})$$

Taking negative log:

$$\mathcal{L}(\theta) = \frac{1}{2} y^\top K_{y,\theta}^{-1} y + \frac{1}{2} \log\det K_{y,\theta} + \frac{n}{2} \log 2\pi \tag{4.28}$$

Breaking this down:

- $\boxed{y^\top K^{-1} y}$ — data fit (how well predictions match y)
- $\boxed{\log\det K}$ — model complexity penalty
- Constant term $\frac{n}{2} \log 2\pi$ — does not depend on θ

Marginal Likelihood for GPs

Drop the constant $\rightarrow$ final objective:

$$\hat{\theta}_{\mathsf{MLE}} = \arg\min_{\theta} \left(\frac{1}{2} y^{\top} K_{y,\theta}^{-1} y + \frac{1}{2} \log \det K_{y,\theta} \right) \tag{4.29}$$

Gradient of Log Marginal Likelihood

To optimize $\mathcal{L}(\theta)$, we need the gradient $\frac{\partial}{\partial \theta_j} \log p(y \mid x, \theta)$.

GPs allow this in closed form:

$$\frac{\partial}{\partial \theta_j} \log p(y \mid x, \theta) = \frac{1}{2} \mathrm{tr}\left[\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - K_{y,\theta}^{-1} \right) \frac{\partial K_{y,\theta}}{\partial \theta_j} \right] \tag{4.30}$$

Where:

- $\boldsymbol{\alpha} = K_{y,\theta}^{-1} y$
- $\frac{\partial K_{y,\theta}}{\partial \theta_j}$ is the derivative of the kernel matrix w.r.t. hyperparameter θ_j
- $\mathrm{tr}(M)$ = trace of matrix M (sum of diagonal elements)

This allows gradient-based optimization (e.g. Adam, SGD).

MAP and Full Bayesian Treatment of θ

MLE: finds the best θ by maximizing $p(y \mid x, \theta)$, when no prior knowledge about θ
MAP: adds a prior over θ:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} \, p(\theta) \cdot p(y \mid x, \theta) \tag{4.31}$$

Taking log:

$$= \arg\min_{\theta} \left(-\log p(\theta) - \log p(y \mid x, \theta) \right) \tag{4.32}$$

Fully Bayesian: integrate out θ

$$p(y^* \mid x^*, \mathcal{D}) = \int \int p(y^* \mid x^*, f) \cdot p(f \mid \mathcal{D}, \theta) \cdot p(\theta) \, df \, d\theta \tag{4.33}$$

Challenge: This integral is intractable in most practical cases.
So we often settle for MLE or MAP, or use variational approximations.

4.5 Why Do We Need Approximations?

Key issue: Computational Cost

- Gaussian Processes require inversion of an $n \times n$ matrix.
- This costs:

$$\mathcal{O}(n^3)$$

- For large datasets ($n \sim 10^4$ or more), this becomes very slow.
- In contrast, Bayesian linear regression has lower cost:

$$\mathcal{O}(nd^2) \quad \text{where } d = \text{input feature dimension}$$

- Therefore, we look for ways to **approximate** a GP while preserving performance.

Next: What happens when optimization isn't even guaranteed to find a global solution?

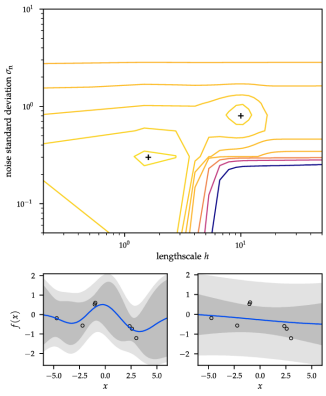
MLL Surface and Two GP Fits

Top plot: Contour plot of negative log marginal likelihood (MLL)

- Axes: Lengthscale h vs noise std σ_n
- Two '+' marks: local optima found during optimization
- Shows that MLL can have multiple optima

Bottom plots: GP regression fits corresponding to the two optima

- Left: small lengthscale, flexible model $\rightarrow$ smooth interpolation
- Right: large noise, rigid model $\rightarrow$ explains variation as noise



Takeaway: Hyperparameter learning is sensitive to initialization and optimization.

4.5.1 Local Methods

- Forward sampling from a GP requires conditioning on all previous observations.
- This becomes computationally expensive as n grows.
- One idea: Only condition on points "near" the test input x.
- For example: keep only points x' such that

$$|k(x, x')| \geq \tau \quad \text{for some threshold } \tau > 0$$

- This "cuts off the tails" of the kernel — treating distant points as independent.

Benefit: Reduces computation.
Caution: If τ is too large, we may lose important long-range correlations.

This is an example of a **sparse approximation** to a GP.

4.5.2 Kernel Function Approximation

Goal: Approximate the kernel $k(x, x')$ directly using a low-dimensional feature map.

Idea:

$$k(x, x') \approx \phi(x)^\top \phi(x') \tag{4.34}$$

Then, apply Bayesian linear regression on $\phi(x)$ instead of GPs.

Computational benefit: Time complexity becomes:

$$\mathcal{O}(nm^2 + m^3)$$

where m is the number of features.

This leads us to **Random Fourier Features (RFF)**, where: - The feature map $\phi(x)$ is constructed from sine and cosine of random projections - Based on Fourier transform theory and Bochner's theorem

Fourier View of Kernels

Euler's identity:

$$e^{ix} = \cos x + i \sin x \tag{4.35}$$

Fourier transform:

$$\hat{f}(\xi) = \int_{\mathbb{R}^d} f(x) e^{-2\pi i \xi^\top x} \, dx \tag{4.36}$$

$$f(x) = \int_{\mathbb{R}^d} \hat{f}(\xi) e^{2\pi i \xi^\top x} \, d\xi \tag{4.37}$$

Stationary kernel as function of $x - x'$:

$$k(x - x') = \int_{\mathbb{R}^d} p(\omega) e^{i\omega^\top (x - x')} d\omega \tag{4.38}$$

Bochner's theorem: $p(\omega)$ is the spectral density of k.

Gaussian kernel's spectral density:

$$p(\omega) = (2h^2\pi)^{d/2} \exp\left(-h^2 \|\omega\|^2 / 2\right) \tag{4.39}$$

Random Fourier Features Approximation

Use Monte Carlo sampling over $\omega \sim p(\omega)$ and $b \sim \text{Unif}[0, 2\pi]$.
Define:

$$z_{\omega,b}(x) = \sqrt{2}\cos(\omega^\top x + b)$$

Then:

$$k(x, x') \approx z(x)^\top z(x') \tag{4.42}$$

$$z(x) = \frac{1}{\sqrt{m}} \begin{bmatrix} z_{\omega^{(1)},b^{(1)}}(x) \\ \vdots \\ z_{\omega^{(m)},b^{(m)}}(x) \end{bmatrix} \tag{4.43}$$

Theorem 4.13: This approximation converges uniformly to the true kernel as $m \to \infty$
Benefit: Fast kernel approximation, scalable GPs!

4.5.3 Inducing Point Methods: Motivation

Problem: Using all n training points in a GP is expensive.

Idea: Use a smaller set of **inducing points** to summarize the full data.

$$U \triangleq \{\bar{x}_1, \ldots, \bar{x}_k\} \quad \text{where } k \ll n$$

We model:

$$u \triangleq \begin{bmatrix} f(\bar{x}_1) & \cdots & f(\bar{x}_k) \end{bmatrix}^\top \quad \sim \mathcal{N}(0, K_{UU})$$

Recover full GP using marginalization:

$$p(f^*, f) = \int p(f^*, f \mid u)\, p(u)\, du \tag{4.45}$$

This lets us reduce computations while preserving most information.

Subset of Regressors

Approximation: Assume f and f^* are conditionally independent given u:

$$p(f^*, f) \approx \int p(f^* \mid u)\, p(f \mid u)\, p(u)\, du \tag{4.46}$$

From Gaussian conditionals:

$$p(f \mid u) \sim \mathcal{N}(K_{AU} K_{UU}^{-1} u,\, K_{AA} - Q_{AA}) \tag{4.47a}$$

$$p(f^* \mid u) \sim \mathcal{N}(K_{*U} K_{UU}^{-1} u,\, K_{**} - Q_{**}) \tag{4.47b}$$

Where $Q_{ab} = K_{aU} K_{UU}^{-1} K_{Ub}$

Subset of Regressors (SoR): keep the mean and drop the variances/covariances

$$q_{\mathsf{SoR}}(f \mid u) = \mathcal{N}(K_{AU} K_{UU}^{-1} u, 0) \tag{4.48a}$$

$$q_{\mathsf{SoR}}(f^* \mid u) = \mathcal{N}(K_{*U} K_{UU}^{-1} u, 0) \tag{4.48b}$$

Drawback: Since we drop the uncertainties in this approach, therefore, we may unrealistic predictions.

Fully Independent Training Conditional

FITC (Fully Independent Training Conditional):

$$q_{\mathsf{FITC}}(f \mid u) = \mathcal{N}(K_{AU}K_{UU}^{-1}u,\, \mathsf{diag}(K_{AA} - Q_{AA})) \tag{4.49a}$$

$$q_{\mathsf{FITC}}(f^* \mid u) = \mathcal{N}(K_{*U}K_{UU}^{-1}u,\, \mathsf{diag}(K_{**} - Q_{**})) \tag{4.49b}$$

Summary:

- SoR: keeps mean, discards all variance
- FITC: keeps mean + variances, ignores off-diagonal covariances

Graph: SoR vs FITC

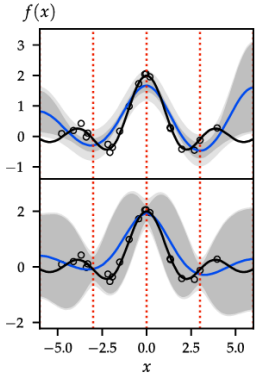


Figure 4.15: Comparison of SoR (top) and FITC (bottom). The inducing points $\boldsymbol{u}$ are shown as vertical dotted red lines. The noise-free true function is shown in black and the mean of the Gaussian process is shown in blue.

Discussion: GP Approximations in Practice

- GPs are flexible non-parametric models that reason over functions.
- But exact inference becomes costly with large datasets.

This chapter introduced several ways to approximate GPs:

- **Local methods:** Use only nearby data points
- **Kernel approximations:** RFF via Bochner's theorem
- **Inducing points:** Summarize data through select locations

Next: We explore how to combine these probabilistic ideas with scalable models.

Thank you!

THE UNIVERSITY OF NEW MEXICO

OPTIMIZATION AND ESTIMATION LAB

ONE.UNM.EDU