Probabilistic Graphical Models
Lecture 20: Gaussian Processes

Andrew Gordon Wilson

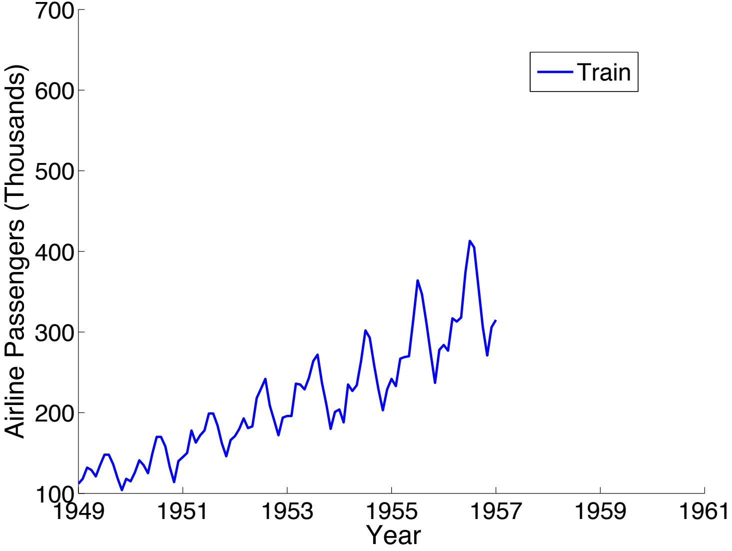
www.cs.cmu.edu/~andrewgw
Carnegie Mellon University

What is Machine Learning?

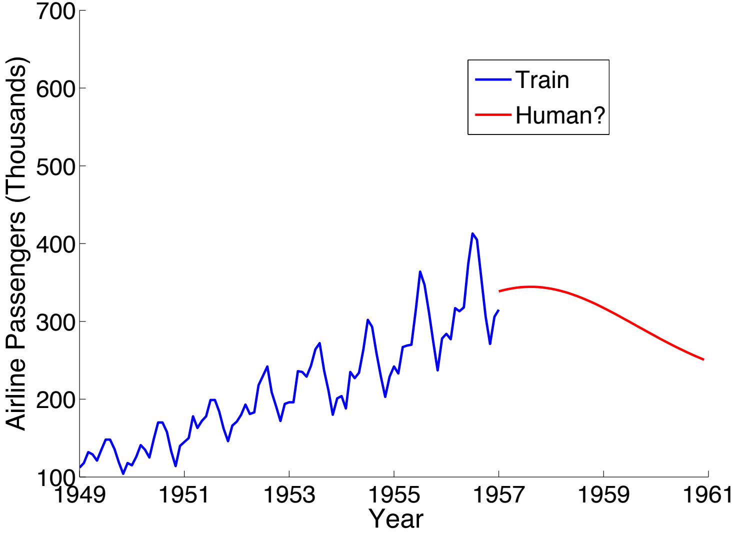
- Machine learning algorithms adapt with data versus having fixed decision rules.
- Machine learning aims not only to equip people with tools to analyse data, but to create algorithms which can learn and make decisions without human intervention.[1,2]
- In order for a model to automatically learn and make decisions, it must be able to discover patterns and extrapolate those patterns to new situations.

[1] E.g. , N.D. Lawrence (2010), "What is Machine Learning?"
[2] T.M. Mitchell (2006), "What is Machine Learning and Where Is it Headed?"

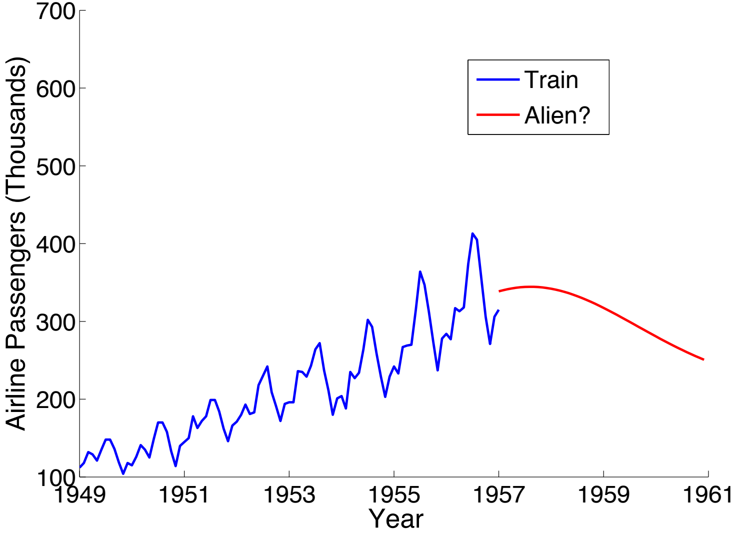
Function Learning Example



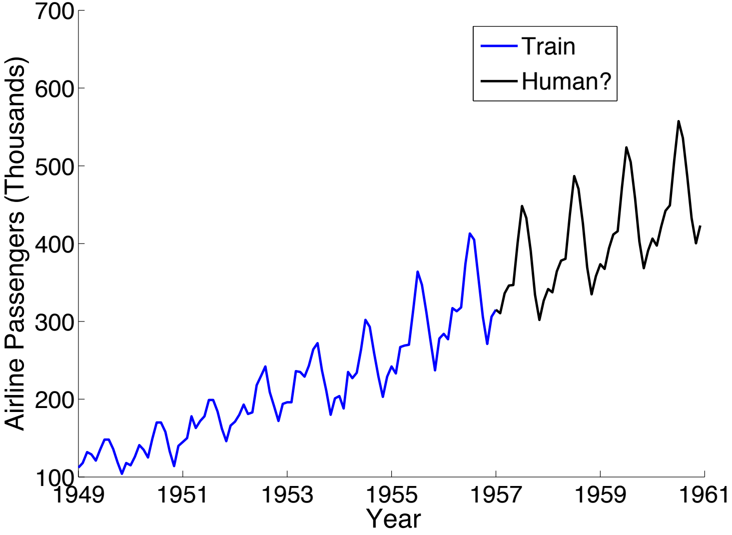
Function Learning Example



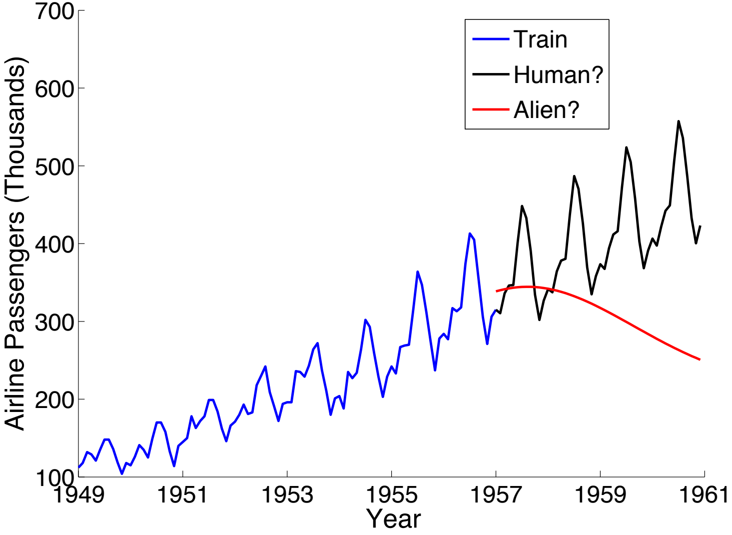
Function Learning Example



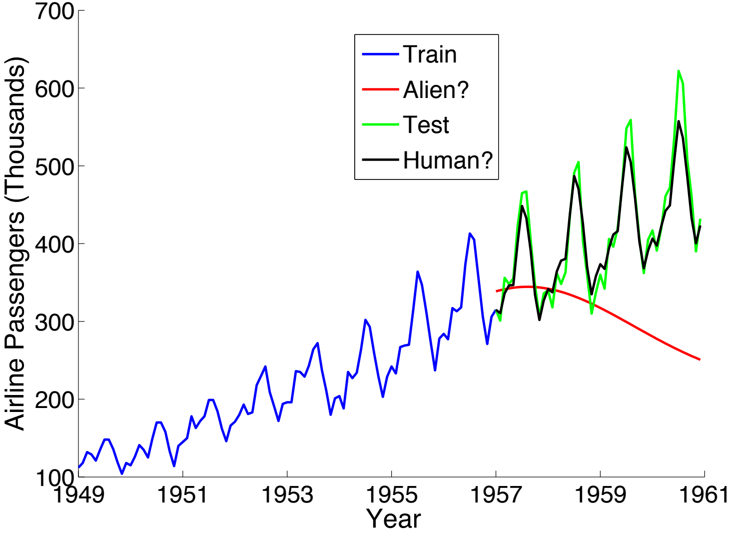
Function Learning Example



Function Learning Example



Function Learning Example



Building an Intelligent Model

The ability for a model to learn from data depends on its:

1. Support: what solutions we think are a priori possible.
2. Inductive biases: what solutions we think are a priori likely.

- Examples: Function Learning, Character Recognition
- Human ability to make remarkable generalisations from data could derive from an expressive prior combined with Bayesian inference.

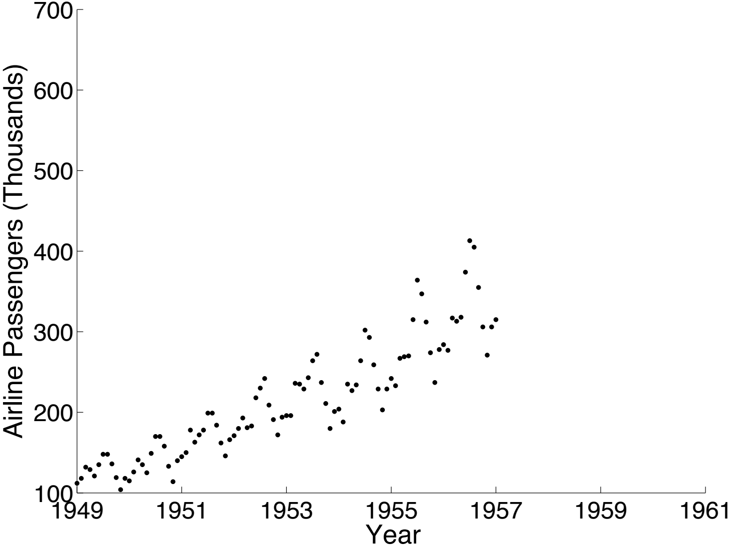
Basic Regression Problem

- Training set of N targets (observations) $\boldsymbol{y} = (y(x_1), \ldots, y(x_N))^{\mathrm{T}}$.
- Observations evaluated at inputs $X = (x_1, \ldots, x_N)^{\mathrm{T}}$.
- Want to predict the value of $y(x_*)$ at a test input x_*.

For example: Given CO_2 concentrations $\boldsymbol{y}$ measured at times X, what will the CO_2 concentration be for $x_* = 2024$, 10 years from now?

Just knowing high school math, what might you try?

Statistics from Scratch



Guess the parametric form of a function that could fit the data

- $f(x, \boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}} x$ [Linear function of $\boldsymbol{w}$ and x]
- $f(x, \boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(x)$ [Linear function of $\boldsymbol{w}$] (Linear Basis Function Model)
- $f(x, \boldsymbol{w}) = g(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(x))$ [Non-linear in x and $\boldsymbol{w}$] (E.g., Neural Network)

$\boldsymbol{\phi}(x)$ is a vector of basis functions. For example, if $\boldsymbol{\phi}(x) = (1, x, x^2)$ and $x \in \mathbb{R}^1$ then $f(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2$ is a quadratic function.

Choose an error measure $E(\boldsymbol{w})$, minimize with respect to $\boldsymbol{w}$

- $E(\boldsymbol{w}) = \sum_{i=1}^{N} [f(x_i, \boldsymbol{w}) - y(x_i)]^2$

A probabilistic approach

We could explicitly account for noise in our model.

- $y(x) = f(x, \boldsymbol{w}) + \epsilon(x)$, where $\epsilon(x)$ is a noise function.

One commonly takes $\epsilon(x) = \mathcal{N}(0, \sigma^2)$ for i.i.d. additive Gaussian noise, in which case

$$p(y(x)|x, \boldsymbol{w}, \sigma^2) = \mathcal{N}(y(x); f(x, \boldsymbol{w}), \sigma^2) \qquad \text{Observation Model} \qquad (1)$$

$$p(\boldsymbol{y}|x, \boldsymbol{w}, \sigma^2) = \prod_{i=1}^{N} \mathcal{N}(y(x_i); f(x_i, \boldsymbol{w}), \sigma^2) \qquad \text{Likelihood} \qquad (2)$$

- Maximize the likelihood of the data $p(\boldsymbol{y}|x, \boldsymbol{w}, \sigma^2)$ with respect to $\sigma^2, \boldsymbol{w}$.

For a Gaussian noise model, this approach will make the same predictions as using a squared loss error function:

$$\log p(\boldsymbol{y}|X, \boldsymbol{w}, \sigma^2) \propto -\frac{1}{2\sigma^2} \sum_{i=1}^{N} [f(x_i, \boldsymbol{w}) - y(x_i)]^2 \qquad (3)$$

Statistics from Scratch

- The probabilistic approach helps us interpret the error measure in a deterministic approach, and gives us a sense of the noise level σ^2.
- Probabilistic methods thus provide an intuitive framework for representing uncertainty, and model development.
- Both approaches are prone to *over-fitting* for flexible $f(x, \boldsymbol{w})$: low error on the training data, high error on the test set.

Regularization

- Use a penalized log likelihood (or error function), such as

$$\log p(\boldsymbol{y}|X, \boldsymbol{w}) \propto \overbrace{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(f(x_i, \boldsymbol{w}) - y(x_i)^2)}^{\text{model fit}} \overbrace{-\lambda \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}}^{\text{complexity penalty}} \quad . \tag{4}$$

- **But how should we define complexity, and how much should we penalize complexity?**
- Can set λ using *cross-validation*.

Bayesian Inference

Bayes' Rule

$$p(a|b) = p(b|a)p(a)/p(b)\,, \qquad p(a|b) \propto p(b|a)p(a)\,. \tag{5}$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}, \quad p(\boldsymbol{w}|\boldsymbol{y}, X, \sigma^2) = \frac{p(\boldsymbol{y}|X, \boldsymbol{w}, \sigma^2)p(\boldsymbol{w})}{p(\boldsymbol{y}|X, \sigma^2)}\,. \tag{6}$$

Predictive Distribution

$$p(y|x_*, \boldsymbol{y}, X) = \int p(y|x_*, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{y}, X)d\boldsymbol{w}\,. \tag{7}$$

- Average of infinitely many models weighted by their posterior probabilities.
- No over-fitting, automatically calibrated complexity.
- Typically more interested in distribution over functions than in parameters $\boldsymbol{w}$.

Representing Uncertainty

Different types of uncertainty:

- Uncertainty through lack of knowledge
- Intrinsic uncertainty; e.g., radioactive decay.

Uncertainty through lack of knowledge can seem like intrinsic uncertainty (e.g., rolling dice).

Regardless of whether or not the universe is deterministic – whether there is some underlying true answer – we will always have uncertainty. We can represent this belief using probability distributions (Bayesian methods, probabilistic modelling).

Parametric Regression Review

Deterministic

$$E(\boldsymbol{w}) = \sum_{i=1}^{N} (f(x_i, \boldsymbol{w}) - y_i)^2 \,. \tag{8}$$

Maximum Likelihood

$$p(y(x)|x, \boldsymbol{w}) = \mathcal{N}(y(x); f(x, \boldsymbol{w}), \sigma_n^2) \,, \tag{9}$$

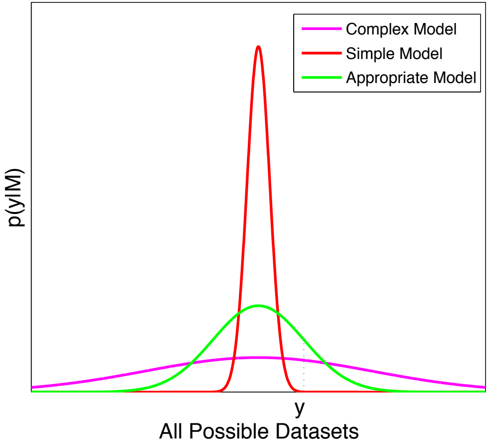
$$p(\boldsymbol{y}|X, \boldsymbol{w}) = \prod_{i=1}^{N} \mathcal{N}(y(x_i); f(x_i, \boldsymbol{w}), \sigma_n^2) \,. \tag{10}$$

Bayesian

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}, \quad p(\boldsymbol{w}|\boldsymbol{y}, X) = \frac{p(\boldsymbol{y}|X, \boldsymbol{w}) p(\boldsymbol{w})}{p(\boldsymbol{y}|X)} \,. \tag{11}$$

Model Selection and Marginal Likelihood

$$p(\boldsymbol{y}|\mathcal{M}_1, X) = \int p(\boldsymbol{y}|f_1(x, \boldsymbol{w}))p(\boldsymbol{w})d\boldsymbol{w} \tag{13}$$



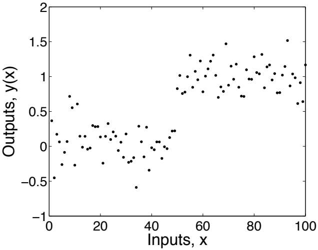
Blackboard: Examples of Occam's Razor in Everyday Inferences

For further reading, see MacKay (2003) textbook, *Information Theory, Inference, and Learning Algorithms*.

-1, 3, 7, 11, ??, ??

- H_1: the sequence is an arithmetic progression, add *n*, where *n* is an integer.
- H_2: the sequence is generated by a cubic function of the form $cx^3 + dx^2 + e$, where c, d, and e are fractions. $(-\frac{1}{11}x^3 + \frac{9}{11}x^2 + \frac{23}{11})$

Model Selection



Observations $y(x)$. Assume $p(y(x)|f(x)) \sim \mathcal{N}(y(x); f(x), \sigma^2)$. Consider polynomials of different orders. As always, observations are out of the chosen model class! Which model should we choose?

$$f_0(x) = a_0\,, \tag{14}$$

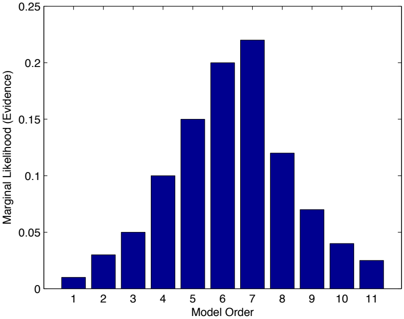
$$f_1(x) = a_0 + a_1 x\,, \tag{15}$$

$$f_2(x) = a_0 + a_1 x + a_2 x^2\,, \tag{16}$$

$$\vdots \tag{17}$$

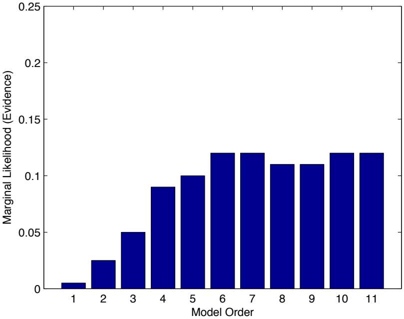
$$f_J(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_J x^J\,. \tag{18}$$

Model Selection: Occam's Hill



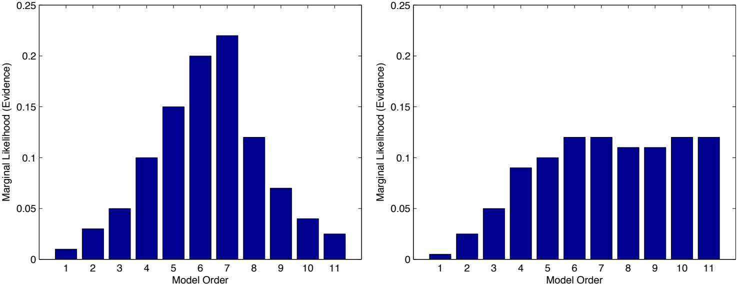
Marginal likelihood (evidence) as a function of model order, using an isotropic prior $p(a) = \mathcal{N}(0, \sigma^2 I)$.

Model Selection: Occam's Asymptote



Marginal likelihood (evidence) as a function of model order, using an anisotropic prior $p(a_i) = \mathcal{N}(0, \gamma^{-i})$, with γ learned from the data.

Occam's Razor



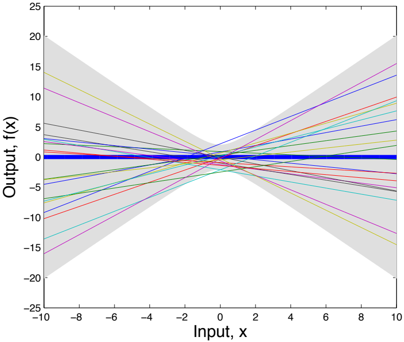
(a) Isotropic Gaussian Prior (b) Anisotropic Gaussian Prior

For further reading, see Rasmussen and Ghahramani (2001) (*Occam's Razor*) and Kass and Raftery (1995) (*Bayes Factors*)

Consider the simple linear model,

$$f(x) = a_0 + a_1 x\,, \tag{19}$$

$$a_0, a_1 \sim \mathcal{N}(0, 1)\,. \tag{20}$$



Linear Models

We are interested in the induced distribution over functions, not the parameters...
Let's characterise the properties of these functions directly:

$$
\begin{aligned}
f(x|a_0, a_1) &= a_0 + a_1 x\,, \qquad a_0, a_1 \sim \mathcal{N}(0,1)\,. && (21)\\
\mathbb{E}[f(x)] &= \mathbb{E}[a_0] + \mathbb{E}[a_1]x = 0\,. && (22)\\
\mathrm{cov}[f(x_b), f(x_c)] &= \mathbb{E}[f(x_b)f(x_c)] - \mathbb{E}[f(x_b)]\mathbb{E}[f(x_c)] && (23)\\
&= \mathbb{E}[a_0^2 + a_0 a_1 (x_b + x_c) + a_1^2 x_b x_c] - 0 && (24)\\
&= \mathbb{E}[a_0^2] + \mathbb{E}[a_1^2 x_b x_c] + \mathbb{E}[a_0 a_1 (x_b + x_c)] && (25)\\
&= 1 + x_b x_c + 0 && (26)\\
&= 1 + x_b x_c\,. && (27)
\end{aligned}
$$

Therefore any collection of values has a joint Gaussian distribution

$$[f(x_1),\ldots,f(x_N)] \sim \mathcal{N}(0,K)\,, \tag{28}$$
$$K_{ij} = \operatorname{cov}(f(x_i),f(x_j)) = k(x_i,x_j) = 1 + x_b x_c\,. \tag{29}$$

By definition, $f(x)$ is a Gaussian process.

Definition

A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution. We write $f(x) \sim \mathcal{GP}(m,k)$ to mean

$$[f(x_1),\ldots,f(x_N)] \sim \mathcal{N}(\boldsymbol{\mu},K) \tag{30}$$
$$\boldsymbol{\mu}_i = m(x_i) \tag{31}$$
$$K_{ij} = k(x_i,x_j)\,, \tag{32}$$

for any collection of input values $x_1,\ldots,x_N$. In other words, f is a GP with mean function $m(x)$ and *covariance kernel* $k(x_i,x_j)$.

Linear Basis Function Models

Model Specification

$$f(x, \boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(x) \tag{33}$$

$$p(\boldsymbol{w}) = \mathcal{N}(0, \Sigma_w) \tag{34}$$

Moments of Induced Distribution over Functions

$$\mathbb{E}[f(x, \boldsymbol{w})] = m(x) = \mathbb{E}[\boldsymbol{w}^{\mathrm{T}}]\boldsymbol{\phi}(x) = 0 \tag{35}$$

$$\mathrm{cov}(f(x_i), f(x_j)) = k(x_i, x_j) = \mathbb{E}[f(x_i)f(x_j)] - \mathbb{E}[f(x_i)]\mathbb{E}[f(x_j)] \tag{36}$$

$$= \boldsymbol{\phi}(x_i)^{\mathrm{T}}\mathbb{E}[\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}]\boldsymbol{\phi}(x_j) - 0 \tag{37}$$

$$= \boldsymbol{\phi}(x_i)^{\mathrm{T}}\Sigma_w\boldsymbol{\phi}(x_j) \tag{38}$$

- $f(x, \boldsymbol{w})$ is a Gaussian process, $f(x) \sim \mathcal{N}(m, k)$ with mean function $m(x) = 0$ and covariance kernel $k(x_i, x_j) = \boldsymbol{\phi}(x_i)^{\mathrm{T}}\Sigma_w\boldsymbol{\phi}(x_j)$.
- The entire basis function model of Eqs. (33) and (34) is encapsulated as a distribution over functions with kernel $k(x, x')$.

Gaussian Processes

- We are ultimately more interested in – and have stronger intuitions about – the *functions* that model our data than weights $\boldsymbol{w}$ in a parametric model, and we can express those intuitions using a covariance kernel.
- The kernel controls the support and inductive biases of our model, and thus its ability to generalise.

Example: RBF Kernel

$$k_{\mathrm{RBF}}(x, x') = \mathrm{cov}(f(x), f(x')) = a^2 \exp(-\frac{||x - x'||^2}{2\ell^2}) \tag{39}$$

- Far and above the most popular kernel.
- Expresses the intuition that function values at nearby inputs are more correlated than function values at far away inputs.
- The kernel *hyperparameters* a and ℓ control amplitudes and wiggliness of these functions.
- GPs with an RBF kernel have large support and are *universal approximators*.

Sampling from a GP with an RBF Kernel

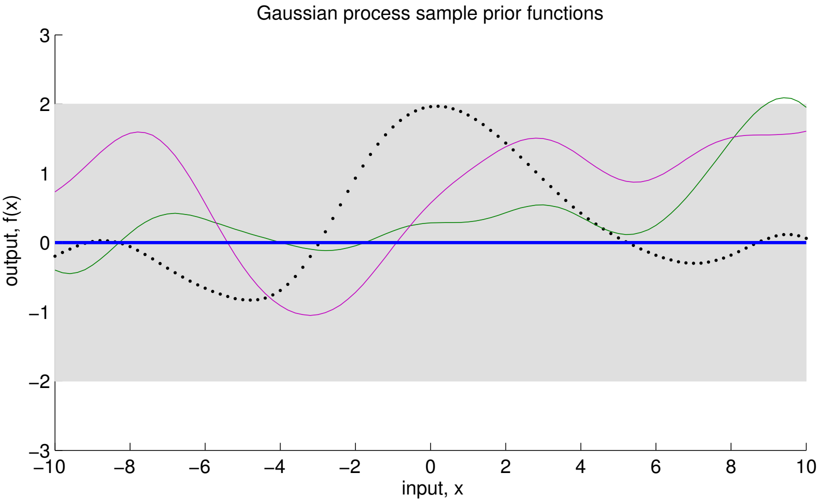
```
x = [-10:0.2:10]';  % inputs (where we query the GP)
N = numel(x);  % number of inputs
K = zeros(N,N);  % covariance matrix

% very inefficient way of creating K in Matlab
for i=1:N
  for j=1:N
      K(i,j) = k_rbf(x(i),x(j));
  end
end

K = K + 1e-6*eye(N);  % add jitter for conditioning
CK = chol(K);
f = CK'*randn(N,1);  % draws from N(0,K)

plot(x,f);
```

Samples from a GP with an RBF Kernel



1D RBF Kernel with Different Length-scales

$$k_{\mathrm{RBF}}(x, x') = \mathrm{cov}(f(x), f(x')) = a^2 \exp\left(-\frac{||x - x'||^2}{2\ell^2}\right) \tag{40}$$

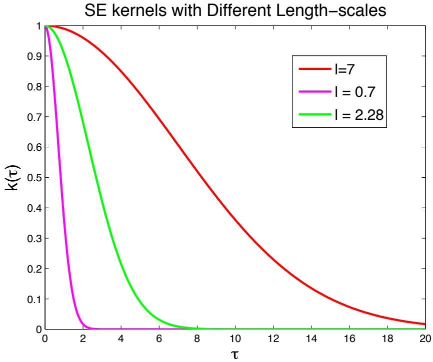
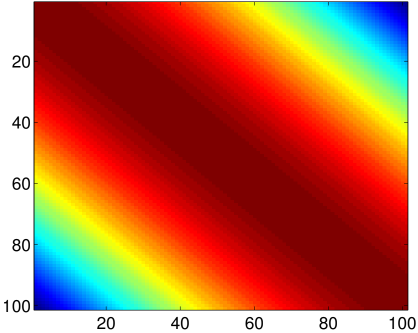


Figure: SE kernels with different length-scales, as a function of $\tau = x - x'$.

RBF Kernel Covariance Matrix

$$k_{\mathrm{RBF}}(x, x') = \mathrm{cov}(f(x), f(x')) = a^2 \exp(-\frac{||x - x'||^2}{2\ell^2}) \tag{41}$$

The covariance matrix K for ordered inputs on a 1D grid. $K_{ij} = k_{\mathrm{RBF}}(x_i, x_j)$.



Gaussian Process Inference

- Observed noisy data $\boldsymbol{y} = (y(x_1), \ldots, y(x_N))^{\mathrm{T}}$ at input locations X.
- Start with the standard regression assumption: $\mathcal{N}(y(x); f(x), \sigma^2)$.
- Place a Gaussian process distribution over noise free functions $f(x) \sim \mathcal{GP}(0, k_\theta)$. The kernel k is parametrized by θ.
- Infer $p(\boldsymbol{f}_*|\boldsymbol{y}, X, X_*)$ for the noise free function f evaluated at test points X_*.

Joint distribution

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_\theta(X, X) + \sigma^2 I & K_\theta(X, X_*) \\ K_\theta(X_*, X) & K_\theta(X_*, X_*) \end{bmatrix}\right). \tag{42}$$

Conditional predictive distribution

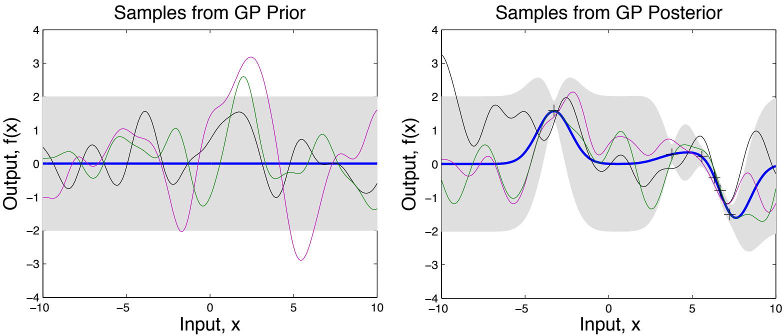
$$\boldsymbol{f}_*|X_*, X, \boldsymbol{y}, \boldsymbol{\theta} \sim \mathcal{N}(\bar{\boldsymbol{f}}_*, \operatorname{cov}(\boldsymbol{f}_*)), \tag{43}$$

$$\bar{\boldsymbol{f}}_* = K_\theta(X_*, X)[K_\theta(X, X) + \sigma^2 I]^{-1}\boldsymbol{y}, \tag{44}$$

$$\operatorname{cov}(\boldsymbol{f}_*) = K_\theta(X_*, X_*) - K_\theta(X_*, X)[K_\theta(X, X) + \sigma^2 I]^{-1}K_\theta(X, X_*). \tag{45}$$

Inference using an RBF kernel

- Specify $f(x) \sim \mathcal{GP}(0, k)$.
- Choose $k_{\text{RBF}}(x, x') = a_0^2 \exp(-\frac{||x-x'||^2}{2\ell_0^2})$. Choose values for a_0 and ℓ_0.
- Observe data, look at the prior and posterior over functions.

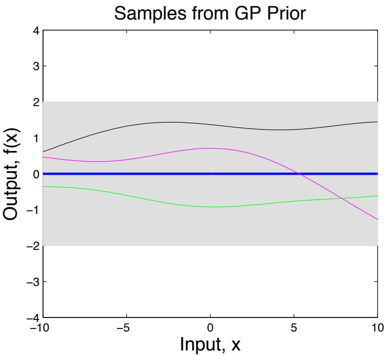


(a) (b)

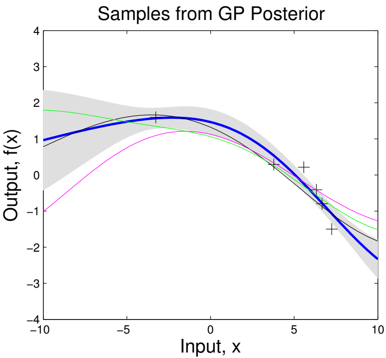
- Does something look strange about these functions?

Inference using an RBF kernel

Increase the length-scale ℓ.



(a)

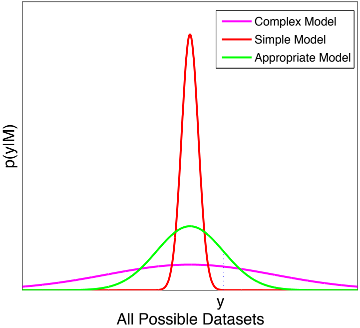


(b)

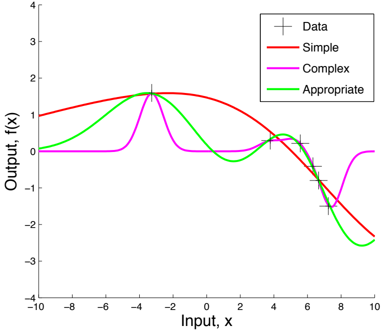
$$p(\mathcal{M}_i|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\mathcal{M}_i)p(\mathcal{M}_i)}{p(\boldsymbol{y})} \tag{46}$$

We can write the *evidence* of the model as

$$p(\boldsymbol{y}|\mathcal{M}_i) = \int p(\boldsymbol{y}|\boldsymbol{f}, \mathcal{M}_i)p(\boldsymbol{f})d\boldsymbol{f}\ , \tag{47}$$



(a)



(b)

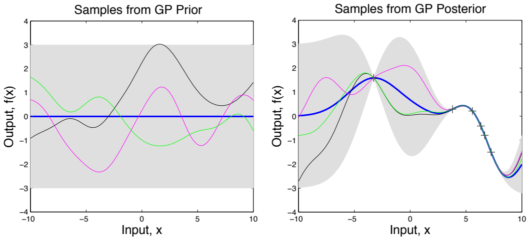
Learning and Model Selection

- We can integrate away the entire Gaussian process $f(x)$ to obtain the marginal likelihood, as a function of kernel hyperparameters θ alone.

$$p(\boldsymbol{y}|\boldsymbol{\theta}, X) = \int p(\boldsymbol{y}|\boldsymbol{f}, X)p(\boldsymbol{f}|\boldsymbol{\theta}, X)d\boldsymbol{f} \,. \tag{48}$$

$$\log p(\boldsymbol{y}|\boldsymbol{\theta}, X) = \overbrace{-\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\boldsymbol{y}}^{\text{model fit}} - \overbrace{\frac{1}{2}\log|K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}} - \frac{N}{2}\log(2\pi) \,. \tag{49}$$

- An extremely powerful mechanism for kernel learning.



Learning and Model Selection

- A fully Bayesian treatment would integrate away kernel hyperparameters θ.

$$p(\boldsymbol{f}_* | X_*, X, \boldsymbol{y}) = \int p(\boldsymbol{f}_* | X_*, X, \boldsymbol{y}, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \boldsymbol{y}) d\boldsymbol{\theta} \tag{50}$$

- For example, we could specify a prior $p(\boldsymbol{\theta})$, use MCMC to take J samples from $p(\boldsymbol{\theta}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})$, and then find

$$p(\boldsymbol{f}_* | X_*, X, \boldsymbol{y}) \approx \frac{1}{J} \sum_{i=1}^{J} p(\boldsymbol{f}_* | X_*, X, \boldsymbol{y}, \boldsymbol{\theta}^{(i)}) \,, \quad \boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta}|\boldsymbol{y}) \,. \tag{51}$$

- If we have a non-Gaussian noise model, and thus cannot integrate away $\boldsymbol{f}$, the strong dependencies between Gaussian process $\boldsymbol{f}$ and hyperparameters $\boldsymbol{\theta}$ make sampling extremely difficult. In my experience, the most effective solution is to use a deterministic approximation for the posterior $p(\boldsymbol{f}|\boldsymbol{y})$ which enables one to work with an approximate marginal likelihood.

Gaussian Process Covariance Kernels

Let $\tau = x - x'$:

$$k_{\mathrm{SE}}(\tau) = \exp(-0.5\tau^2/\ell^2) \tag{52}$$

$$k_{\mathrm{MA}}(\tau) = a(1 + \frac{\sqrt{3}\tau}{\ell})\exp(-\frac{\sqrt{3}\tau}{\ell}) \tag{53}$$

$$k_{\mathrm{RQ}}(\tau) = (1 + \frac{\tau^2}{2\,\alpha\,\ell^2})^{-\alpha} \tag{54}$$

$$k_{\mathrm{PE}}(\tau) = \exp(-2\sin^2(\pi\,\tau\,\omega)/\ell^2) \tag{55}$$

Inference and Learning

1. Learning: Optimize marginal likelihood,

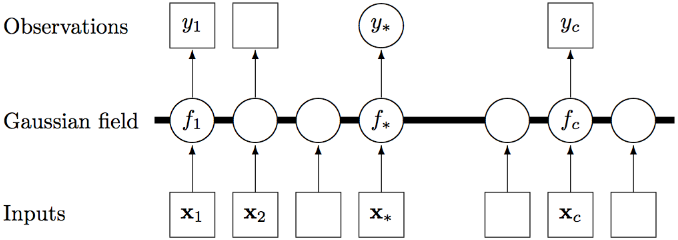
$$\log p(\boldsymbol{y}|\boldsymbol{\theta}, X) = \overbrace{-\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\boldsymbol{y}}^{\text{model fit}} - \overbrace{\frac{1}{2}\log|K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}} - \frac{N}{2}\log(2\pi)\,,$$

   with respect to kernel hyperparameters $\boldsymbol{\theta}$.

2. Inference: Conditioned on kernel hyperparameters $\boldsymbol{\theta}$, form the predictive distribution for test inputs X_*:

$$\begin{aligned}
\boldsymbol{f}_*|X_*, X, \boldsymbol{y}, \boldsymbol{\theta} &\sim \mathcal{N}(\bar{\boldsymbol{f}}_*, \mathrm{cov}(\boldsymbol{f}_*))\,,\\
\bar{\boldsymbol{f}}_* &= K_\theta(X_*, X)[K_\theta(X, X) + \sigma^2 I]^{-1}\boldsymbol{y}\,,\\
\mathrm{cov}(\boldsymbol{f}_*) &= K_\theta(X_*, X_*) - K_\theta(X_*, X)[K_\theta(X, X) + \sigma^2 I]^{-1}K_\theta(X, X_*)\,.
\end{aligned}$$

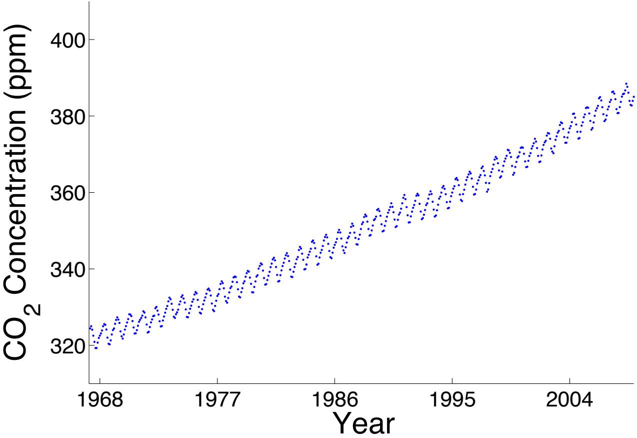
Gaussian process graphical model



- Squared are observed, circles are latent, the thick bar is a set of fully connected nodes.
- Each y_i is conditionally independent given f_i.
- Because of the marginalization property of a GP, addition of further inputs x_* and unobserved targets y_* does not change the distribution of any other variables.

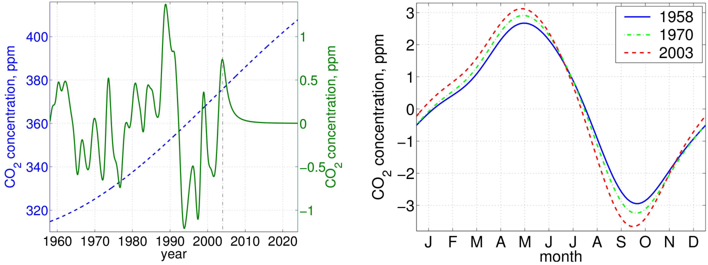
Figure from GPML, Rasmussen and Williams (2006)

Worked Example: Combining Kernels, CO_2 Data

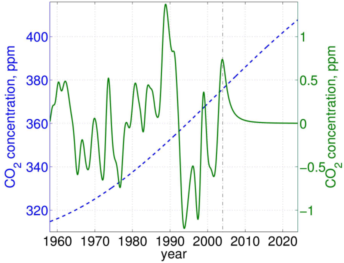


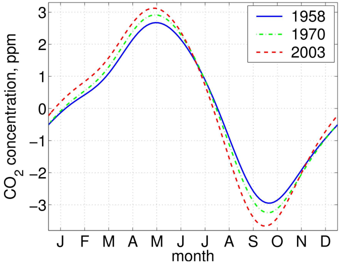
Example from Rasmussen and Williams (2006), *Gaussian Processes for Machine Learning*.

Worked Example: Combining Kernels, CO_2 Data



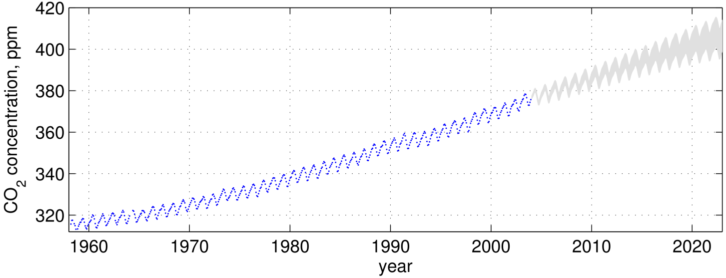
Worked Example: Combining Kernels, CO_2 Data





- Long rising trend: $k_1(x_p, x_q) = \theta_1^2 \exp\left(-\frac{(x_p - x_q)^2}{2\theta_2^2}\right)$
- Quasi-periodic seasonal changes: $k_2(x_p, x_q) =$ $k_{\text{RBF}}(x_p, x_q) k_{\text{PER}}(x_p, x_q) = \theta_3^2 \exp\left(-\frac{(x_p - x_q)}{2\theta_4^2} - \frac{2\sin^2(\pi(x_p - x_q))}{\theta_5^2}\right)$
- Multi-scale medium term irregularities: $k_3(x_p, x_q) = \theta_6^2 \left(1 + \frac{(x_p - x_q)^2}{2\theta_8 \theta_7^2}\right)^{-\theta_8}$
- Correlated and i.i.d. noise: $k_4(x_p, x_q) = \theta_9^2 \exp\left(-\frac{(x_p - x_q)^2}{2\theta_{10}^2}\right) + \theta_{11}^2 \delta_{pq}$
- $k_{\text{total}}(x_p, x_q) = k_1(x_p, x_q) + k_2(x_p, x_q) + k_3(x_p, x_q) + k_4(x_p, x_q)$

Worked Example: Combining Kernels, CO_2 Data



- Hand crafted a kernel combination to perform extrapolation
- Confidence in the extrapolation is high (suggests that model is well specified).
- Can interpret the learned kernel hyperparameters θ to learn information about our dataset.
- A lot of the interesting pattern recognition has been done by a human in this example. We would like to completely automate this modelling procedure.

Non-Gaussian Likelihoods

We can no longer analytically integrate away the Gaussian process. But we can use a simple Monte carlo sum:

$$
\begin{aligned}
p(f_*|\boldsymbol{y}, X, x_*) &= \int p(f_*|\boldsymbol{f}, x_*)p(\boldsymbol{f}|\boldsymbol{y})d\boldsymbol{f} \\
&\approx \frac{1}{J}\sum_{j=1}^{J} p(f_*|\boldsymbol{f}^{(j)}, x_*)\,, \quad \boldsymbol{f}^{(j)} \sim p(\boldsymbol{f}|\boldsymbol{y})
\end{aligned}
$$

But how do we sample from $p(\boldsymbol{f}|\boldsymbol{y})$?

Non-Gaussian Likelihoods

We can no longer analytically integrate away the Gaussian process. But we can use a simple Monte carlo sum:

$$\begin{aligned} p(f_*|\boldsymbol{y}, X, x_*) &= \int p(f_*|\boldsymbol{f}, x_*)p(\boldsymbol{f}|\boldsymbol{y})d\boldsymbol{f} \\ &\approx \frac{1}{J}\sum_{j=1}^{J} p(f_*|\boldsymbol{f}^{(j)}, x_*)\,, \quad \boldsymbol{f}^{(j)} \sim p(\boldsymbol{f}|\boldsymbol{y}) \end{aligned}$$

But how do we sample from $p(\boldsymbol{f}|\boldsymbol{y})$?

Elliptical slice sampling. Murray et. al. AISTATS 2010.

Non-Gaussian Likelihoods

But what about hyperparameters? It's easy to implement Gibbs sampling:

$$p(\boldsymbol{f}|\boldsymbol{y}, \theta) \propto p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{\theta}) \tag{56}$$

$$p(\boldsymbol{\theta}|\boldsymbol{f}, \boldsymbol{y}) \propto p(\boldsymbol{f}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \,. \tag{57}$$

But this won't work because of strong correlations between $\boldsymbol{f}$ and $\boldsymbol{\theta}$.

Non-Gaussian Likelihoods

But what about hyperparameters? It's easy to implement Gibbs sampling:

$$p(\boldsymbol{f}|\boldsymbol{y}, \theta) \propto p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{\theta}) \tag{58}$$

$$p(\boldsymbol{\theta}|\boldsymbol{f}, \boldsymbol{y}) \propto p(\boldsymbol{f}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,. \tag{59}$$

But this won't work because of strong correlations between $\boldsymbol{f}$ and $\boldsymbol{\theta}$.

- Transform into a *whitened* space, $\boldsymbol{f} = L\boldsymbol{\nu}$, and sample from $\boldsymbol{\nu}$ and $\boldsymbol{\theta}$, which decouples correlations.

Non-Gaussian Likelihoods

But what about hyperparameters? It's easy to implement Gibbs sampling:

$$p(\boldsymbol{f}|\boldsymbol{y},\theta) \propto p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{\theta}) \tag{60}$$

$$p(\boldsymbol{\theta}|\boldsymbol{f},\boldsymbol{y}) \propto p(\boldsymbol{f}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,. \tag{61}$$

But this won't work because of strong correlations between $\boldsymbol{f}$ and $\boldsymbol{\theta}$.

- Transform into a *whitened* space, $\boldsymbol{f} = L\boldsymbol{\nu}$, and sample from $\boldsymbol{\nu}$ and $\boldsymbol{\theta}$, which decouples correlations.
- Use a deterministic approach to approximately integrate away $\boldsymbol{f}$ to access a marginal likelihood, conditioned only on kernel hyperparameters $\boldsymbol{\theta}$:

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{\theta})d\boldsymbol{f} \tag{62}$$

- The *Laplace approximation*, for example, approximates $p(\boldsymbol{f}|\boldsymbol{y})$ as a Gaussian.

Readings for Next Time

- C. Rasmussen and C. Williams, GPML, Ch. 4, 5
- Y. Saatchi, PhD Thesis, 2011. Chapter 5
- J. Candela and C.E. Rasmussen, A unifying view of sparse approximation Gaussian process regression, JMLR 2005.
- A.G. Wilson and R.P. Adams. Gaussian process kernels for pattern discovery and extrapolation, ICML 2013.