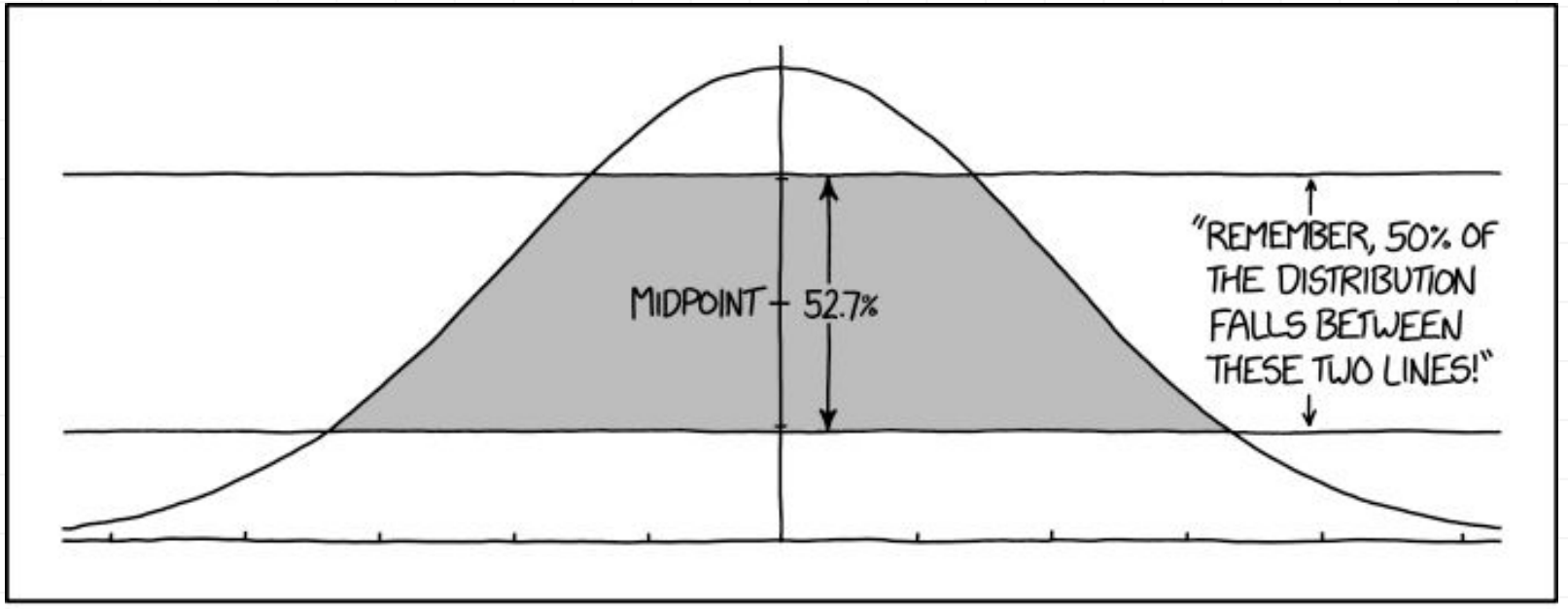
https://xkcd.com/2118/



HOW TO ANNOY A STATISTICIAN

Announcements

Course schedule updated.

In person lectures - PSYC G8
Week 8 Wed - Gaussian Process 2
Week 10 Wed - Guest lecture: ML for Cyber security

- No tutorial in week 7 – assignment 1 walk-through during drop-in, will be recorded.
- No tutorial in week 8 – enjoy quiz 2 :)

- Video assignment released, due in week 12.
- Assignment 2 out soon.
- Quiz 2 released, 9 multiple choice questions.
- Assignment 1 raw grades out, regrade-request open on Gradescope for 2 weeks.

Thank You for anonymous survey input

IML → SML

Student A: Math too hard!

Student B: “Content is too easy. Does not feel challenging enough to be a 4000 level course. Make assessment more difficult, and material more in depth.”

? skipped derivations / proofs

Thanks. Let’s clarify some expectations for derivations / proofs.
Things that are important and appearing for the first time are generally covered; when a proof is skipped it usually means that we expect students to be able to follow the overall logic, and the model design arguments without the proof.

? motivating the material

Motivations tend to be mathematical in this course. We’re happy to help in a number of ways.

? which parts are important

Good question. We’ll try to clarify + reemphasize this in the intro/outro slides. If still unclear, do ask questions during class Q&A, during tuts, or on piazza.

Gaussian Processes - Regression

Bishop, Chap 6.4 (6.4.1-6.4.3)
GP book
http://gaussianprocess.org/gpml/chapters/ Chap 1, 2.1, 2.2, 2.5

Motivation

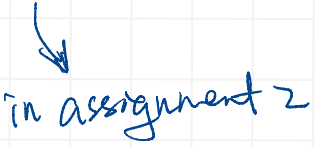
Defining Gaussian Processes (GP)

Kernel functions, sampling

GP regression

GP regression - predictive distribution

Sampling algorithm and computational costs



Bayesian Joint Inversions for the Exploration of Earth Resources

Alistair Reid[1], **Simon O'Callaghan**[1], **Edwin V. Bonilla**[1], **Lachlan McCalman**[1], **Tim Rawling**[2] and **Fabio Ramos**[3]

1. NICTA, 2. University of Melbourne, 3. University of Sydney

{ alistair.reid, simon.ocallaghan, edwin.bonilla, lachlan.mccalman }@nicta.com.au

tim.rawling@unimelb.edu.au, f.ramos@acfr.usyd.edu.au

In a geological inversion problem, properties such as temperature, conductivity, density, magnetic susceptibility and permeability are inferred from related observations such as gravity, magnetics and seismic reflexion. ... In this paper we formulate geophysical inversion as a machine learning problem, and propose an approach based on *Gaussian processes regression* that naturally provides both a predictive distribution over the inverted quantities and a principled method to fuse different types of observations. We apply our method to a real dataset from South Australia containing gravity and drill-hole data with the goal of characterizing rock densities for geothermal target exploration, and also to simulated validation data involving gravity, drill-hole and magnetic observations.

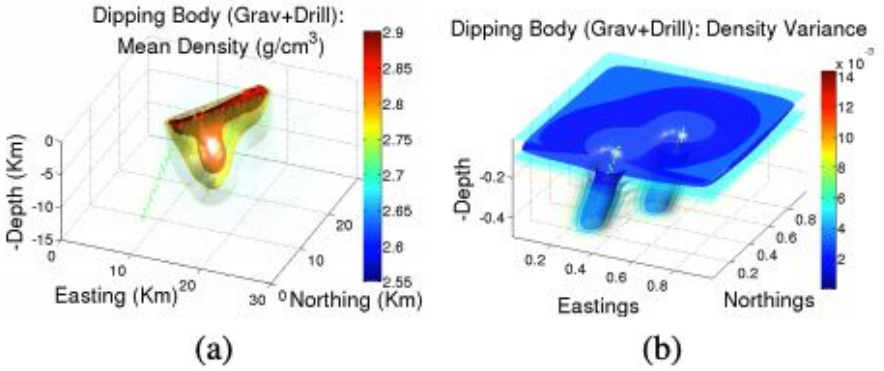


Figure 5: Outputs of the Gaussian process inversion algorithm after fusing gravity and drill observations of the dipping body.

Two ways to learn a function (regression)

[GPbook, intro]

Motivating scenarios

- SARCOS robotic arm
  - x: (7 joint positions, 7 joint velocities, 7 joint accelerations)
  - y: 7 joint torques
  - Learning: y ~ f(x)
  - Prediction: compute the torque needed to move the arm along a given trajectory
- Predict the yield of a chemical plant (y), given temperature, pressure, amount of catalyst, etc (x)
  - Hmm, we don't know a functional form of y (in either input or kernel space)
  - With full uncertainty estimate in y

Representation choices

- Restrict the class of functions we consider, e.g. linear — we know how to do this :)
- Give prior probability to every possible function — huh?

Un-countably infinite number of functions
Un-countably infinite number of input points (e.g. R, R^2, ...)

Rely on parameter of the function, which can be a function itself
"Pretend" that it's a very long vector

Prediction function for regression

$$\mathbf{w}^\star = (\lambda \mathbf{I} + \mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top \mathbf{t}$$

normal eq.

Prediction function for new **x**

$$y(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}^\star = \boldsymbol{\phi}(\mathbf{x})^\top (\lambda \mathbf{I} + \mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top \mathbf{t}$$

Kernelized version

$$\mathbf{w} = \mathbf{\Phi}^\top \mathbf{a}$$

$$\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}. \tag{6.8}$$

$\Phi\Phi^\top$

$$y(\mathbf{x}) = \mathbf{w}^\mathrm{T} \boldsymbol{\phi}(\mathbf{x}) = \mathbf{a}^\mathrm{T} \mathbf{\Phi} \boldsymbol{\phi}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\mathrm{T} (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t} \tag{6.9}$$

point estimate of y "Bayesian" view desires

$P(y_{N+1} \mid x_{N+1}, x_{1:N}, y_{1:N})$

↘ expressed in K

What is the Bayesian version of this?

Bayesian linear regression (isotropic prior)

GP book 2.1

def. "weight-space"

if

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \quad (3.10)$$

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \quad (3.52)$$

then

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N) \quad (3.49)$$

regression prob.

learning: weights etc.

$$\mathbf{m}_N = \beta \mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} \quad (3.53)$$

$$\mathbf{S}_N^{-1} = \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}. \quad (3.54)$$

Prediction function for new **x**

$$p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x})) \quad (3.58)$$

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}(\mathbf{x}). \quad (3.59)$$

prediction: (x_{N+1}, t_{N+1}) ?, $P(t_{N+1}(\cdots))$

Is there a kernelized version of this? – can I have both kernels and prior

Motivation

In week 6, we talked about kernels in non-probabilistic settings:

$$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{t}$$

$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b = \sum_{n=1}^{N} a_n \, t_n \, k(\mathbf{x}, \mathbf{x}_n) + b$$

Can one use kernels in a probabilistic setting?

Starting again from linear regression ...

"noiseless"

fixed

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) \tag{6.49}$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \tag{6.50}$$

matrix form

fixed

gaussian

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{w} \tag{6.51}$$

$$\Phi_{nk} = \phi_k(\mathbf{x}_n)$$

Claim: Y is Gaussian ... only it's mean and covariance matters.

$$\mathbb{E}[\mathbf{y}] = \boldsymbol{\Phi}\mathbb{E}[\mathbf{w}] = \mathbf{0} \tag{6.52}$$

$\alpha^{-1}\mathbf{I}$

$$\mathrm{cov}[\mathbf{y}] = \mathbb{E}\left[\mathbf{y}\mathbf{y}^{\mathrm{T}}\right] = \boldsymbol{\Phi}\mathbb{E}\left[\mathbf{w}\mathbf{w}^{\mathrm{T}}\right]\boldsymbol{\Phi}^{\mathrm{T}} = \frac{1}{\alpha}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} = \mathbf{K} \tag{6.53}$$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K})$$

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m) \tag{6.54}$$

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\alpha}\Phi\,\Phi^{T})$$

stochastic proc · $P(y(t) \mid \theta)$

Gaussian process (GP)

A *Gaussian Process* is defined as a probability distribution over functions y(x) such that the set of values of y(x) evaluated at an arbitrary set of points $x_1, \ldots, x_N$ jointly have a Gaussian distribution.

- More generally, *a stochastic process* y(x) is specified by giving the joint probability distribution for any finite set of values $y(x_1), \ldots, y(x_N)$ in a consistent manner.
- Common choice: set mean to zero, due to no prior knowledge of y(x) $E(y) = \vec{0}$
- The joint distribution of any $y_1, \ldots y_N$ Fully specified by their second-order statistics
- Input x in 2-D – *Gaussian random field* (also called Kriging in statistics)

x in 1-D – easy to visualise

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) \tag{6.49}$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \tag{6.50}$$

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{w} \tag{6.51}$$

$\vec{y} \sim N(\vec{0}, K)$

Kernel functions in GP

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K})$$

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) \qquad \mathbf{y} = \boldsymbol{\Phi}\mathbf{w}$$

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \tag{6.50}$$

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_m) \tag{6.54}$$

Implicitly defined by ϕ(x)

One can also explicitly define a kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2/2\sigma^2\right) \tag{6.23}$$

$$k(x, x') = \exp\left(-\theta\,|x - x'|\right) \tag{6.56}$$

$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\top}\mathbf{x}'$	linear kernel
$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$	stationary kernel
$k(\mathbf{x}, \mathbf{x}') = k(\|\mathbf{x} - \mathbf{x}'\|)$	homogeneous kernel

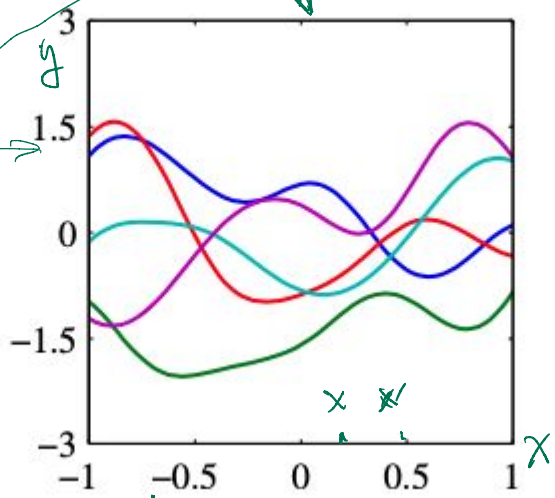
The whole heap of kernel construction tricks apply ...

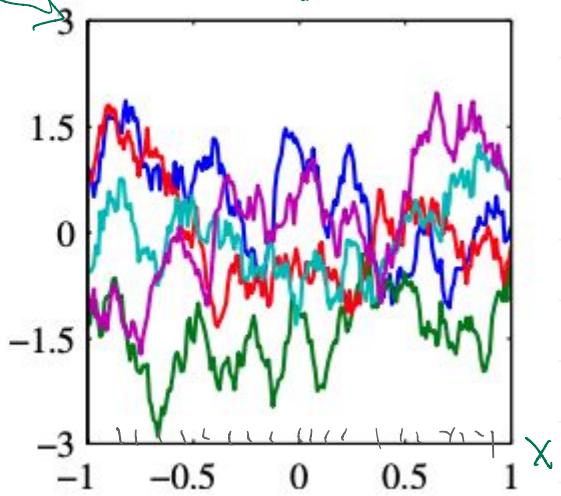
changes more slowly.

short-term fluctuations

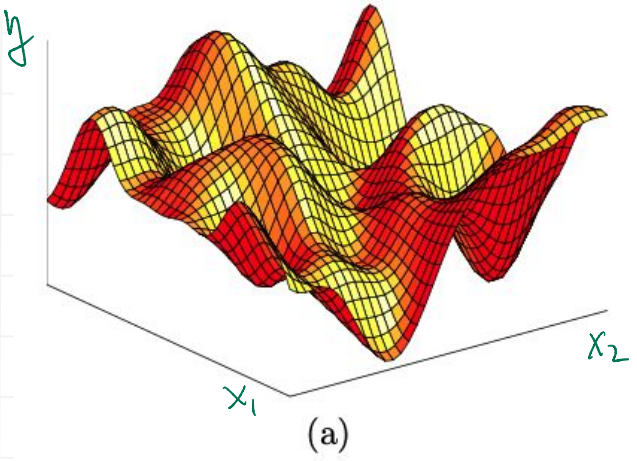
Figure 6.4 Samples from Gaussian processes for a 'Gaussian' kernel (left) and an exponential kernel (right).

$k(x, x')$ $\quad \mathrm{cov}(y(x), y(x'))$





[GP book, Fig 1.2(a)]



$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2\right) \tag{6.23}$$

$$k(x, x') = \exp\left(-\theta\,|x - x'|\right) \tag{6.56}$$

$\vec{y} \sim N(\vec{0}, K)$

$\{x_1, x_2, \ldots, x_n\}$

$K \in \mathbb{R}^{n \times n}$.

Q: How to sample from a GP?
How are these figures produced?

[GP book, Sec A.2] http://gaussianprocess.org/gpml/chapters/RWA.pdf

generating multivariate Gaussian samples

To generate samples $\mathbf{x} \sim \mathcal{N}(\mathbf{m}, K)$ with arbitrary mean $\mathbf{m}$ and covariance matrix K using a scalar Gaussian generator (which is readily available in many programming environments) we proceed as follows: first, compute the Cholesky decomposition (also known as the "matrix square root") L of the positive definite symmetric covariance matrix $K = LL^\top$, where L is a lower triangular matrix, see section A.4. Then generate $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, I)$ by multiple separate calls to the scalar Gaussian generator. Compute $\mathbf{x} = \mathbf{m} + L\mathbf{u}$, which has the desired distribution with mean $\mathbf{m}$ and covariance $L\mathbb{E}[\mathbf{u}\mathbf{u}^\top]L^\top = LL^\top = K$ (by the independence of the elements of $\mathbf{u}$).

numpy.random.standard_normal

In practice it may be necessary to add a small multiple of the identity matrix ϵI to the covariance matrix for numerical reasons. This is because the eigenvalues of the matrix K can decay very rapidly (see section 4.3.1 for a closely related analytical result) and without this stabilization the Cholesky decomposition fails. The effect on the generated samples is to add additional independent noise of variance ϵ. From the context ϵ can usually be chosen to have inconsequential effects on the samples, while ensuring numerical stability.

numpy.random.multivariate_normal

$\begin{bmatrix} x_1'' \\ x_2'' \end{bmatrix} = U \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}$

rotate by matrix U

$x_1' \sim \alpha_1 x_1$

$x_2' \sim \alpha_2 x_2$

easy!

$x_1 \sim N(0,1)$

$x_2 \sim N(0,1)$

Intuition:

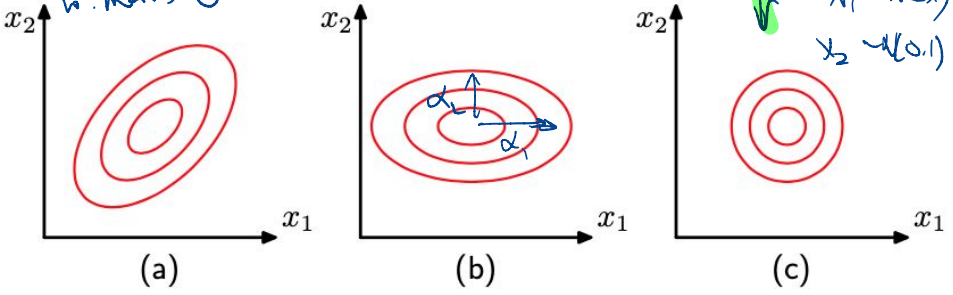


Figure 2.8 Contours of constant probability density for a Gaussian distribution in two dimensions in which the covariance matrix is (a) of general form, (b) diagonal, in which the elliptical contours are aligned with the coordinate axes, and (c) proportional to the identity matrix, in which the contours are concentric circles.

Why Cholesky rather than eigen decomposition?

faster to compute!

$O(n^3)$ < Cholesky, eigen val, inverse

Cost of some matrix factorizations and decompositions. A is $n \times n$, except for QR and SVD, where it is $m \times n$ ($m \geq n$).

Factorization/decomposition	Number of flops
LU factorization with partial pivoting ($PA = LU$)	$2n^3/3$
LU factorization with partial pivoting of upper Hessenberg matrix ($PA = LU$)	n^2
Cholesky factorization ($A = R^*R$)	$n^3/3$
Householder QR factorization ($A = QR$)	$2n^2(m - n/3)$ for R; $4(m^2n - mn^2 + n^3/3)$ for $m \times m$ Q; $2n^2(m - n/3)$ for $m \times n$ Q; $2np(2m - n)$ for QB with $m \times p$ B and Q held in factored form.
SVD[a] ($A = P\Sigma Q^*$)	$14mn^2 + 8n^3$ ($P(:, 1:n)$, Σ, and Q)[b] $6mn^2 + 20n^3$ ($P(:, 1:n)$, Σ, and Q)[c]
Hessenberg decomposition ($A = QHQ^*$)	$14n^3/3$ (Q and H), $10n^3/3$ (H only)
Schur decomposition[a] ($A = QTQ^*$)	$25n^3$ (Q and T), $10n^3$ (T only)
For Hermitian A:	
Tridiagonal reduction ($A = QTQ^*$)	$8n^3/3$ (Q and T), $4n^3/3$ (T only)
Spectral decomposition ($A = QDQ^*$)	$9n^3$ (Q and D), $4n^3/3$ (D only)

[a] These costs are estimates taken from Golub and Van Loan [224, 1996].
[b] Golub–Reinsch SVD.
[c] Golub–Reinsch SVD with preliminary QR factorization.

https://math.stackexchange.com/questions/2840755/what-is-the-computation-time-of-lu-cholesky-and-qr-decomposition

Gaussian Processes - Regression

Motivation

Defining Gaussian Processes (GP)

Kernel functions, sampling

GP regression

GP regression - predictive distribution

Sampling algorithm and computational costs

Bishop, Chap 6.4 (6.4.1-6.4.3)
GP book
http://gaussianprocess.org/gpml/chapters/ Chap 1, 2.1, 2.2, 2.5

GP for regression (w noise)

"noiseless"

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\alpha}\Phi\,\Phi^T)$$

$\sim \mathcal{N}(0, K)$

$$y_n = y(\mathbf{x}_n) \qquad t_n = y_n + \epsilon_n \qquad \epsilon_n \sim \mathcal{N}(0, \beta^{-1}) \tag{6.57}$$

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1}) \tag{6.58}$$

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N) \tag{6.59}$$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}). \tag{6.60}$$

$C = \begin{bmatrix} k_{11}+\beta^{-1} & k_{12} & \cdots \\ k_{21} & \cdots & \\ \cdots & & k_{nn}+\beta^{-1} \end{bmatrix}$

$E(y)=0 \quad E(\epsilon)=0$

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})\,\mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \tag{6.61}$$

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}. \tag{6.62}$$

$$\mathbf{t} \sim \mathcal{N}(\mathbf{0}, \frac{1}{\beta}I_N + K)$$

GP for regression

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}). \tag{6.60}$$

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})\,\mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \tag{6.61}$$

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}. \tag{6.62}$$

training

Let's explicitly define a kernel

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp\left\{-\frac{\theta_1}{2}\|\mathbf{x}_n - \mathbf{x}_m\|^2\right\} + \theta_2 + \theta_3 \mathbf{x}_n^{\mathrm{T}}\mathbf{x}_m. \tag{6.63}$$

sq-exponential const linear

We covered representation and "training" (implicit) in GP.

via C

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp\left\{-\frac{\theta_1}{2}\|\mathbf{x}_n - \mathbf{x}_m\|^2\right\} + \theta_2 + \theta_3 \mathbf{x}_n^{\mathrm{T}} \mathbf{x}_m. \tag{6.63}$$

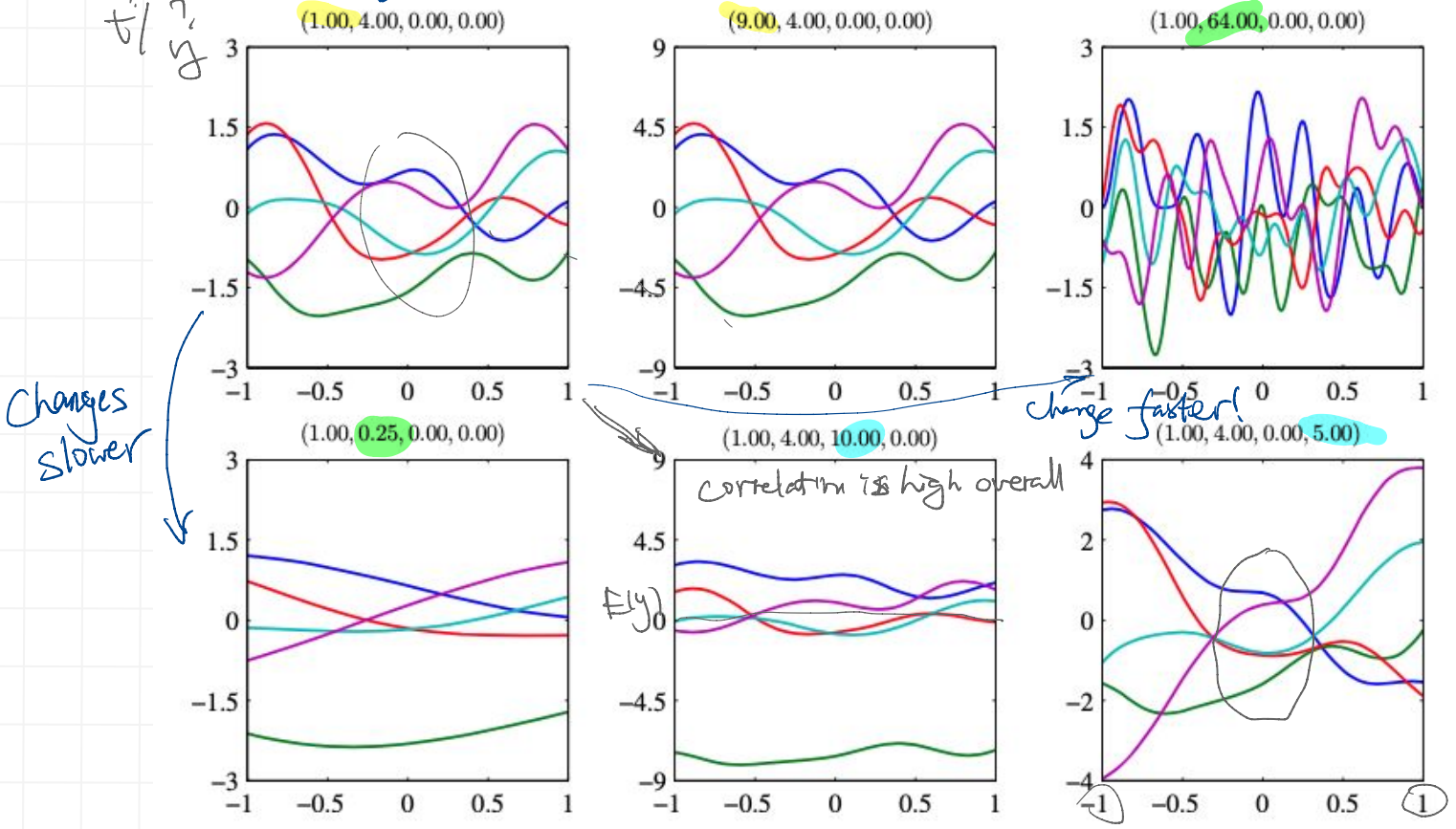


Figure 6.5 Samples from a Gaussian process prior defined by the covariance function (6.63). The title above each plot denotes $(\theta_0, \theta_1, \theta_2, \theta_3)$.

Predictive distribution $p(t_{N+1}|\mathbf{t}_N)$

$p(\vec{t}_N) \sim \mathcal{N}(0, C_N)$

$$p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1}) \tag{6.64}$$

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^{\mathrm{T}} & c \end{pmatrix} \tag{6.65}$$

$K + \beta^{-1} I_N$ → kernel val of test pt x_{N+1}

vector $\mathbf{k}$ has elements $k(\mathbf{x}_n, \mathbf{x}_{N+1})$

$$c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$$

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b) \tag{2.81}$$

$$\boldsymbol{\Sigma}_{a|b} = \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba}. \tag{2.82}$$

The conditional mean and variance of t_{N+1} given $t_{1:N}$, expressed as a function of x_{N+1}

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} \tag{6.66}$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}. \tag{6.67}$$

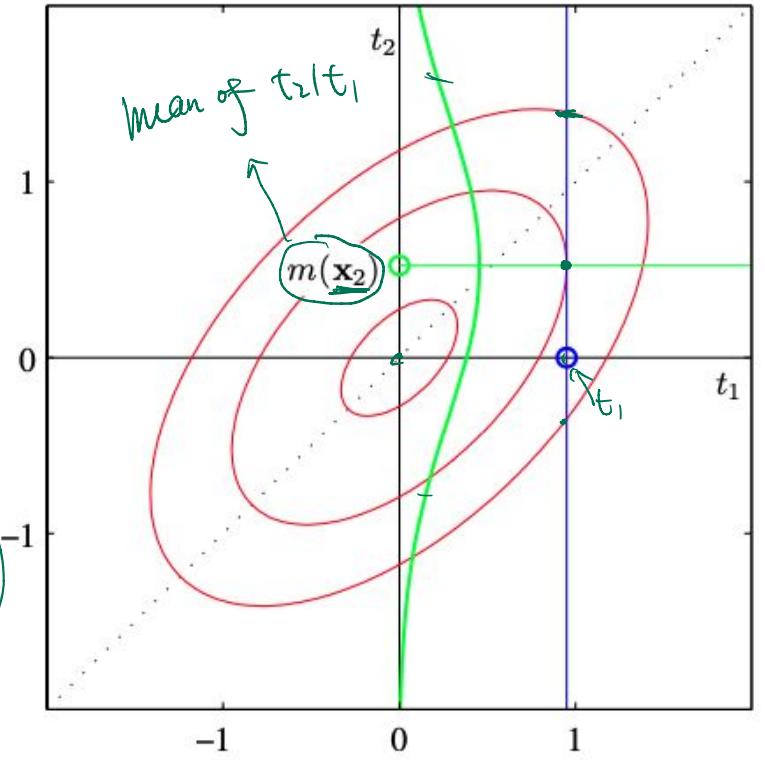
$m(x_{N+1})$ of t_{N+1} · "training" · k: test input, training inputs · c: variance reduction due to "training"

k^T

$$m(\mathbf{x}_{N+1}) = \sum_{n=1}^{N} a_n k(\mathbf{x}_n, \mathbf{x}_{N+1}) \tag{6.68}$$

where a_n is the n^{th} component of $\mathbf{C}_N^{-1}\mathbf{t}$.

Figure 6.7 Illustration of the mechanism of Gaussian process regression for the case of one training point and one test point, in which the red ellipses show contours of the joint distribution $p(t_1, t_2)$. Here t_1 is the training data point, and conditioning on the value of t_1, corresponding to the vertical blue line, we obtain $p(t_2|t_1)$ shown as a function of t_2 by the green curve.



$$p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1})$$

$k(t_1, t_1)$

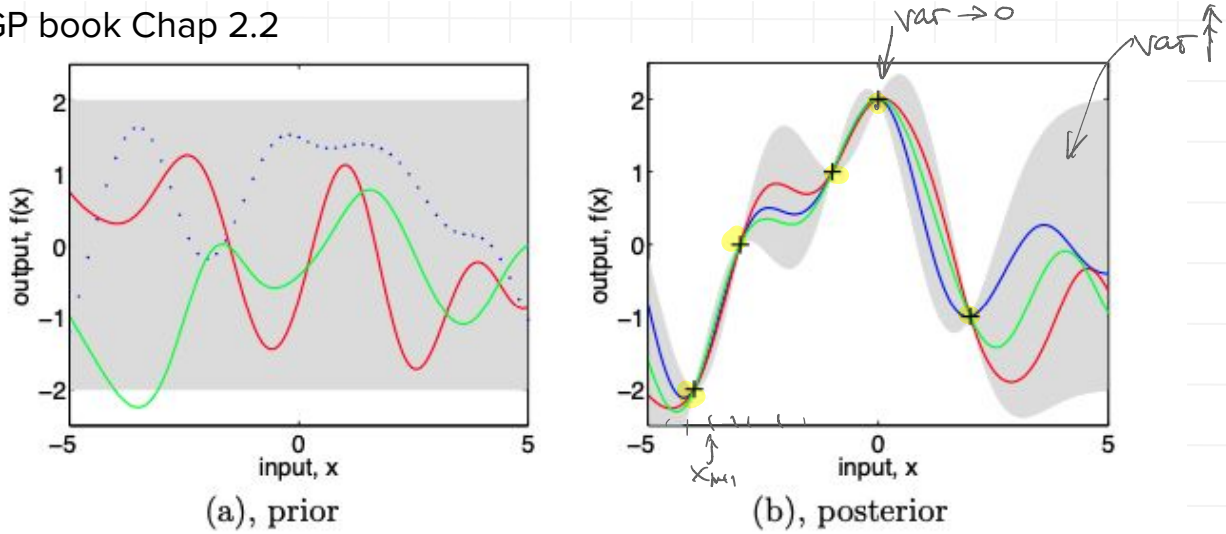
$$p\left(\begin{bmatrix} t_1 \\ t_2 \end{bmatrix}\right) \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{bmatrix} K_{11} & K_{12} \\ K_{12} & K_{22} \end{bmatrix}\right)$$

$p(t_2|t_1)$

Notations

		Bishop book	GP book
"train"	input	x, $\phi(x)$ Φ	x, X $\phi(x)$
	output	$y = \Phi w$ $t = y + \varepsilon$	$f(x) = \phi^T(x)\, w$ $y \sim N(f, \sigma_n^2)$
	Kernel / cov-fn	K, C_N, $\vec{k} = [k(x_{N+1}, x_{1:N})]$	$K(X, X_*)$, $K(X, X)$, $k(x, x')$
"test"	input	x_{N+1}	x_*, X_*
	output	t_{N+1} (scalar) w. $m(x_{N+1})$, $\sigma^2(x_{N+1})$	f_* w. $\overline{f_*}$ $\mathbb{V}[f_*]$
	noise cov	β^{-1}	σ_n^2

GP book Chap 2.2



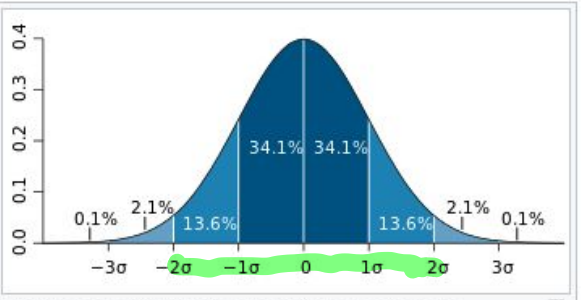
(a), prior (b), posterior

Figure 2.2: Panel (a) shows three functions drawn at random from a GP prior; the dots indicate values of y actually generated; the two other functions have (less correctly) been drawn as lines by joining a large number of evaluated points. Panel (b) shows three random functions drawn from the posterior, i.e. the prior conditioned on the five noise free observations indicated. In both plots the shaded area represents the pointwise mean plus and minus two times the standard deviation for each input value (corresponding to the 95% confidence region), for the prior and posterior respectively.

$$\mathbf{f}_*|X_*, X, \mathbf{f} \sim \mathcal{N}\big(K(X_*, X)K(X, X)^{-1}\mathbf{f}, \\ K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\big). \tag{2.19}$$

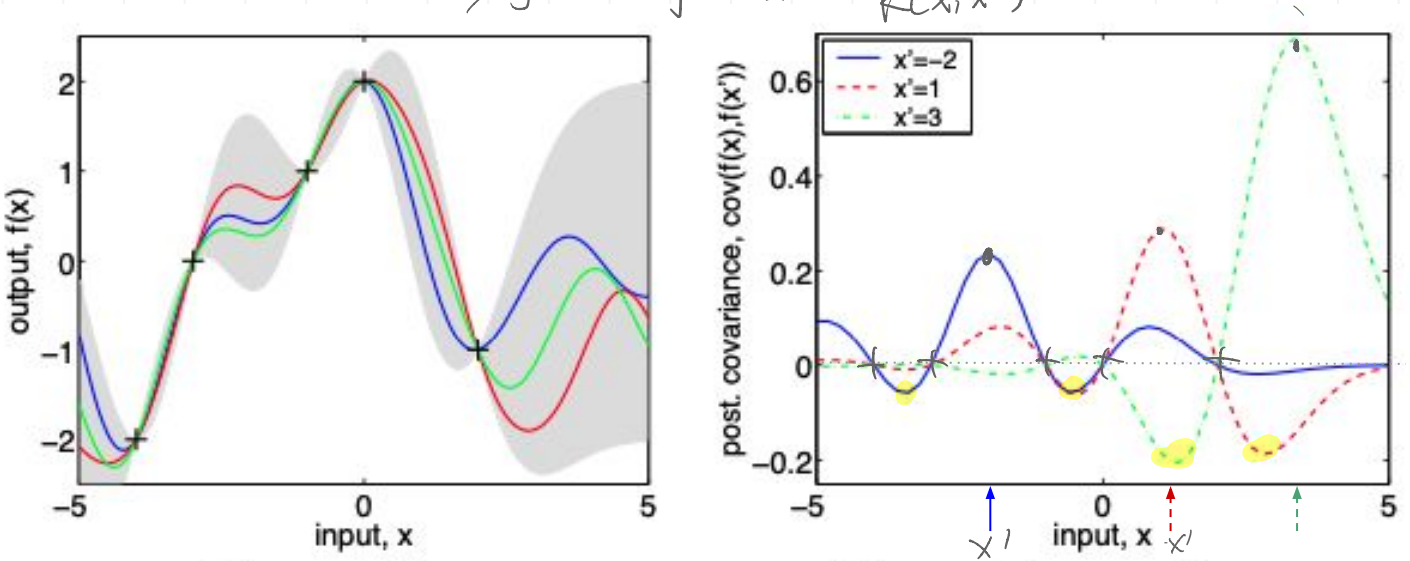
$$m(\mathbf{x}_{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} \tag{6.66}$$
$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}. \tag{6.67}$$

Noise-less: $C_N = K_N$



For the normal distribution, the values less than one standard deviation away from the mean account for 68.27% of the set; while two standard deviations from the mean account for 95.45%; and three standard deviations account for 99.73%.

GP book Chap 2.2



(a), posterior (b), posterior covariance

Figure 2.4: Panel (a) is identical to Figure 2.2(b) showing three random functions drawn from the posterior. Panel (b) shows the posterior *co*-variance between $f(\mathbf{x})$ and $f(\mathbf{x}')$ for the same data for three different values of $\mathbf{x}'$. Note, that the covariance at close points is high, falling to zero at the training points (where there is no variance, since it is a noise-free process), then becomes negative, etc. This happens because if the smooth function happens to be less than the mean on one side of the data point, it tends to exceed the mean on the other side, causing a reversal of the sign of the covariance at the data points. Note for contrast that the *prior* covariance is simply of Gaussian shape and never negative.

GP book Chap 2.2

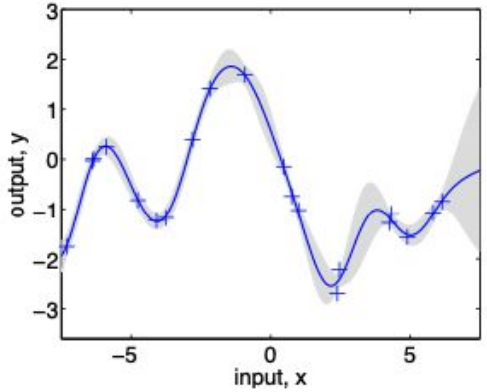
noise var.

$$k_y(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}(x_p - x_q)^2\right) + \sigma_n^2 \delta_{pq}. \qquad (2.31)$$

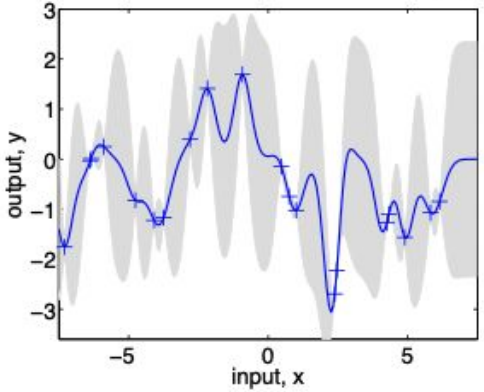
noise low

① Sampling w varying kernel param.

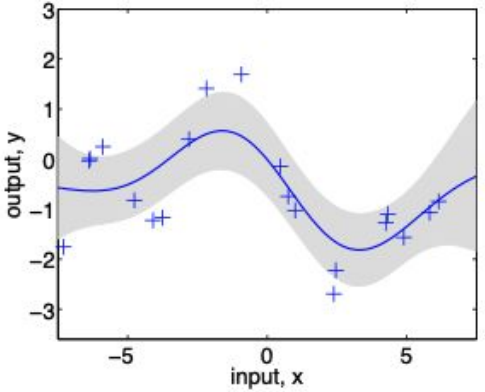
② posterior distribution given input data.



(a), $\ell = 1$



(b), $\ell = 0.3$



(c), $\ell = 3$

Figure 2.5: (a) Data is generated from a GP with hyperparameters $(\ell, \sigma_f, \sigma_n) = (1, 1, 0.1)$, as shown by the + symbols. Using Gaussian process prediction with these hyperparameters we obtain a 95% confidence region for the underlying function f (shown in grey). Panels (b) and (c) again show the 95% confidence region, but this time for hyperparameter values $(0.3, 1.08, 0.00005)$ and $(3.0, 1.16, 0.89)$ respectively.

tion, shown in Figure A.6. The input values $\{x_n\}$ are generated uniformly in range $(0, 1)$, and the corresponding target values $\{t_n\}$ are obtained by first computing the corresponding values of the function $\sin(2\pi x)$, and then adding random noise with a Gaussian distribution having standard deviation 0.3. Various forms of this data set,

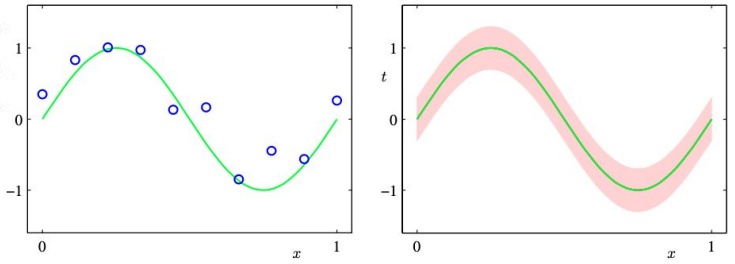
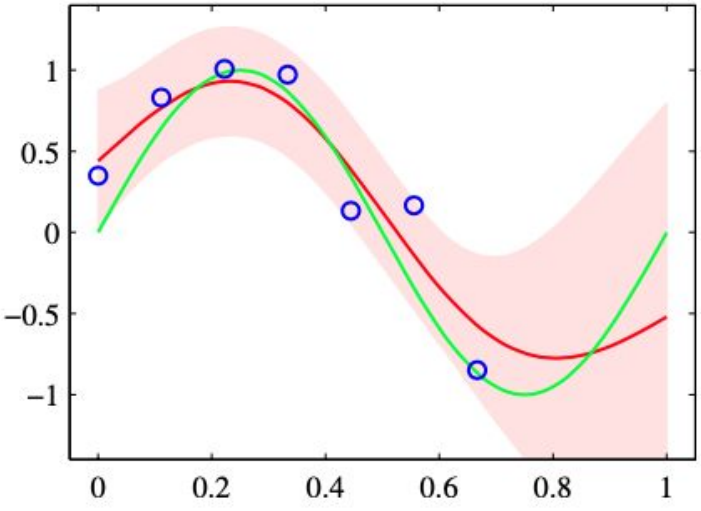


Figure A.6 The left-hand plot shows the synthetic regression data set along with the underlying sinusoidal function from which the data points were generated. The right-hand plot shows the true conditional distribution $p(t|x)$ from which the labels are generated, in which the green curve denotes the mean, and the shaded region spans one standard deviation on each side of the mean.

Figure 6.8 Illustration of Gaussian process regression applied to the sinusoidal data set in Figure A.6 in which the three right-most data points have been omitted. The green curve shows the sinusoidal function from which the data points, shown in blue, are obtained by sampling and addition of Gaussian noise. The red line shows the mean of the Gaussian process predictive distribution, and the shaded region corresponds to plus and minus two standard deviations. Notice how the uncertainty increases in the region to the right of the data points.



$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}. \tag{6.67}$$

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}. \tag{6.62}$$

$$c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$$

[Bishop 6.4]

Implementing

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} \quad (6.66)$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}. \quad (6.67)$$

input: X (inputs), $\mathbf{y}$ (targets), k (covariance function), σ_n^2 (noise level), $\mathbf{x}_*$ (test input)

2: $L := \mathrm{cholesky}(K + \sigma_n^2 I)$ → C_N

$\boldsymbol{\alpha} := L^{\top}\backslash(L\backslash\mathbf{y})$ ← $C_N^{-1}t$

4: $\bar{f}_* := \mathbf{k}_*^{\top}\boldsymbol{\alpha}$ } predictive mean eq. (2.25)

$\mathbf{v} := L\backslash\mathbf{k}_*$

6: $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^{\top}\mathbf{v}$ } predictive variance eq. (2.26)

$\log p(\mathbf{y}|X) := -\frac{1}{2}\mathbf{y}^{\top}\boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2}\log 2\pi$ eq. (2.30)

8: **return**: $\bar{f}_*$ (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood)

Algorithm 2.1: Predictions and log marginal likelihood for Gaussian process regression. The implementation addresses the matrix inversion required by eq. (2.25) and (2.26) using Cholesky factorization, see section A.4. For multiple test cases lines 4-6 are repeated. The log determinant required in eq. (2.30) is computed from the Cholesky factor (for large n it may not be possible to represent the determinant itself). The computational complexity is $n^3/6$ for the Cholesky decomposition in line 2, and $n^2/2$ for solving triangular systems in line 3 and (for each test case) in line 5.

GP book Chap 2.2, A.4

$A = LL^{\mathrm{T}}$, L lower-triangular

To solve $Ax = b$ $C_N^{-1}t$

i.e. $x = A\backslash b$ or $A^{-1}b$ for square A

First solve $Ly=b$, $y = L \backslash b$

Then $L^{\mathrm{T}}x = y$

$x = L^{\mathrm{T}}\backslash y = L^{\mathrm{T}}\backslash (L \backslash b)$

$Lv = k_*$

$v^{\mathrm{T}}v = k^{\mathrm{T}}C_N^{-1}k$

Note also that the determinant of a positive definite symmetric matrix can be calculated efficiently by

$$|A| = \prod_{i=1}^{n} L_{ii}^2, \quad \text{or} \quad \log|A| = 2\sum_{i=1}^{n}\log L_{ii}, \quad (A.18)$$

where L is the Cholesky factor from A.

GP vs Bayesian linear regression

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} \quad (6.66)$$

$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}. \quad (6.67)$$

Bottleneck:
compute C_N^{-1}, $O(N^3)$ - at training time
$O(N^2)$ for each test point

Linear regression w basis functions

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N) \quad (3.49)$$

$$\mathbf{m}_N = \mathbf{S}_N\left(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}\right) \quad (3.50)$$

$$\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}. \quad (3.51)$$

Bottleneck:
compute S_N, $O(M^3)$ - at training time
$O(M^2)$ for each test point

An example application

GP book Chap 2.5

- SARCOS robotic arm
  - x: (7 joint positions, 7 joint velocities, 7 joint accelerations)
  - y: 7 joint torques
  - Learning: y ~ f(x)
  - Prediction: compute the torque needed to move the arm along a given trajectory

Squared exponential covariance function

- Separate length scales for each input dim, σ_f^2, σ_n^2
- Optimise marginal likelihood on subset of data

~49K input-output pairs

- 44,484 for training
- 4,449 for testing

"inverting" Cholesky 40K x 40K

Evaluation metrics

- SMSE - Standardized mean square error
- MSLL - Mean standardised log loss

Method	SMSE	MSLL
LR	0.075	-1.29
RBD	0.104	–
LWPR	0.040	–
GPR	0.011	-2.25

Table 2.1: Test results on the inverse dynamics problem for a number of different methods. The "–" denotes a missing entry, caused by two methods not producing full predictive *distributions*, so MSLL could not be evaluated.

*speed up with approximations, see Table 8.1

Learning hyperparameters

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})\,\mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \tag{6.61}$$

$$\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\ln|\mathbf{C}_N| - \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{t} - \frac{N}{2}\ln(2\pi). \tag{6.69}$$

Gaussian likelihood

$$\frac{\partial}{\partial x}\ln|\mathbf{A}| = \mathrm{Tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\right) \tag{C.22}$$

$$\frac{\partial}{\partial x}\left(\mathbf{A}^{-1}\right) = -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\mathbf{A}^{-1} \tag{C.21}$$

$$\frac{\partial}{\partial \theta_i}\ln p(\mathbf{t}|\boldsymbol{\theta}) = -\frac{1}{2}\mathrm{Tr}\left(\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta_i}\right) + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta_i}\mathbf{C}_N^{-1}\mathbf{t}. \tag{6.70}$$

Gaussian Processes - Regression

Bishop, Chap 6.4 (6.4.1-6.4.3)
GP book
http://gaussianprocess.org/gpml/chapters/ Chap 1, 2.1, 2.2, 2.5

Motivation

Defining Gaussian Processes (GP)

Kernel functions, sampling

GP regression

GP regression - predictive distribution

Sampling algorithm and computational costs