Gaussian Processes - Part I
The Linear Algebra of Inference

Philipp Hennig

MLSS 2013
29 August 2013

Max Planck Institute for Intelligent Systems
Department of Empirical Inference
Tübingen, Germany
Carl Friedrich Gauss (1777–1855)

Paying Tolls with A Bell

\[ f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]
The Gaussian distribution
Multivariate Form

\[ \mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{N/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu)^\top \Sigma^{-1} (x - \mu) \right] \]

- \( x, \mu \in \mathbb{R}^N \), \( \Sigma \in \mathbb{R}^{N \times N} \)
- \( \Sigma \) is positive semidefinite, i.e.
  - \( v^\top \Sigma v \geq 0 \) for all \( v \in \mathbb{R}^N \)
  - Hermitian, all eigenvalues \( \geq 0 \)
Why Gaussian?

an experiment

nothing in the real world is Gaussian (except sums of i.i.d. variables)

But nothing in the real world is linear either!

Gaussians are for inference what linear maps are for algebra.
Closure Under Multiplication

multiple Gaussian factors form a Gaussian

\[ N(x; a, A)N(x; b, B) = N(x; c, C)N(a; b, A + B) \]

\[ C := (A^{-1} + B^{-1})^{-1} \quad c := C(A^{-1}a + B^{-1}b) \]
Closure Under Multiplication

multiple Gaussian factors form a Gaussian

\[ \mathcal{N}(x; a, A)\mathcal{N}(x; b, B) = \mathcal{N}(x; c, C)\mathcal{N}(a; b, A + B) \]

\[ C := (A^{-1} + B^{-1})^{-1} \quad c := C(A^{-1}a + B^{-1}b) \]
Closure Under Multiplication
multiple Gaussian factors form a Gaussian

\[ N(x; a, A)N(x; b, B) = N(x; c, C)N(a; b, A + B) \]
\[ C := (A^{-1} + B^{-1})^{-1} \]
\[ c := C(A^{-1}a + B^{-1}b) \]
Closure under Linear Maps

Linear Maps of Gaussians are Gaussians

\[ p(z) = \mathcal{N}(z; \mu, \Sigma) \]

\[ \Rightarrow p(Az) = \mathcal{N}(Az, A\mu, A\Sigma A^T) \]

Here: \( A = [1, -0.5] \)
Closure under Marginalization

projections of Gaussians are Gaussian

- projection with $A = \begin{pmatrix} 1 & 0 \end{pmatrix}$

$$\int \mathcal{N} \left[ \begin{pmatrix} x \\ y \end{pmatrix} ; \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix} \right] \, dy = \mathcal{N} \left( x; \mu_x, \Sigma_{xx} \right)$$

- this is the sum rule

$$\int p(x, y) \, dy = \int p(y \mid x)p(x) \, dy = p(x)$$

- so every finite-dim Gaussian is a marginal of infinitely many more
Closure under Conditioning

cuts through Gaussians are Gaussians

\[ p(x \mid y) = \frac{p(x, y)}{p(y)} = \mathcal{N}(x; \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y), \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}) \]

- this is the **product rule**
- so Gaussians are closed under the rules of probability
Bayesian Inference
explaining away

\[ p(x) = \mathcal{N}(x; \mu, \Sigma) = \mathcal{N} \left[ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 3^2 & 0 \\ 0 & 3^2 \end{pmatrix} \right] \]
Bayesian Inference
explaining away

\[ p(x) = \mathcal{N}(x; \mu, \Sigma) \]
\[ = \mathcal{N} \left[ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} ; \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} , \begin{pmatrix} 3^2 & 0 \\ 0 & 3^2 \end{pmatrix} \right] \]

\[ p(y \mid x, \sigma) = \mathcal{N}(y; A^T x; \sigma^2) \]
\[ = \mathcal{N} \left[ 6; \begin{pmatrix} 1 & 0.6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} , \sigma^2 \right] \]
Bayesian Inference
explaining away

\[ p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mu, \Sigma) \]
\[ = \mathcal{N} \left[ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; \begin{pmatrix} 1 & 0.5 \\ 0.5 & 3^2 \end{pmatrix} \right] \]

\[ p(y | \mathbf{x}, \sigma) = \mathcal{N}(y; A^\top \mathbf{x}; \sigma^2) \]
\[ = \mathcal{N} \left[ 6; \begin{pmatrix} 1 & 0.6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \sigma^2 \right] \]

\[ p(\mathbf{x} | \sigma^2, y) = \frac{p(\mathbf{x})p(y | \mathbf{x})}{p(\mathbf{x})} \]
Bayesian Inference

explaining away

\[ p(x) = \mathcal{N}(x; \mu, \Sigma) \]
\[ = \mathcal{N} \left[ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} ; \begin{pmatrix} 0.5 \\ 1 \end{pmatrix} , \begin{pmatrix} 3^2 & 0 \\ 0 & 3^2 \end{pmatrix} \right] \]
\[ p(y \mid x, \sigma) = \mathcal{N}(y; A^T x; \sigma^2) \]
\[ = \mathcal{N} \left[ 6; \begin{pmatrix} 1 & 0.6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} , \sigma^2 \right] \]
\[ p(x \mid \sigma^2, y) = \frac{p(x)p(y \mid x)}{p(x)} \]
\[ = \mathcal{N} \left[ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} ; \begin{pmatrix} 3.9 \\ 2.3 \end{pmatrix} , \begin{pmatrix} 3.4 & -3.4 \\ -3.4 & 7.0 \end{pmatrix} \right] \]
What can we do with this?

Linear regression

given $y \in \mathbb{R}^N$, $p(y \mid f)$, what's $f$?
A prior over linear functions

\[ f(x) = w_1 + w_2 x = \phi_x^T w \]

\[ \phi_x = \begin{pmatrix} 1 \\ x \end{pmatrix} \]

\[ p(w) = \mathcal{N}(w; \mu, \Sigma) \]

\[ p(f) = \mathcal{N}(f; \phi_x^T \mu, \phi_x^T \Sigma \phi_x) \]
A prior over linear functions

\[ f(x) = w_1 + w_2 x = \phi_x^T w \]

\[ \phi_x = \begin{pmatrix} 1 \\ x \end{pmatrix} \]

\[ p(w) = \mathcal{N}(w; \mu, \Sigma) \]

\[ p(f) = \mathcal{N}(f; \phi_x^T \mu, \phi_x^T \Sigma \phi_x) \]
The posterior over linear functions

\[ p(y \mid w, \phi_X) = \mathcal{N}(y; \phi_X^T w, \sigma^2 I) \]

\[ p(w \mid y, \phi_X) = \mathcal{N}(w; \mu + \Sigma \phi_X (\phi_X^T \Sigma \phi_X + \sigma^2 I)^{-1} (y - \phi_X^T \mu), \]

\[ \Sigma - \Sigma \phi_X (\phi_X^T \Sigma \phi_X + \sigma^2 I)^{-1} \phi_X^T \Sigma) \phi_x \]
The posterior over linear functions

\[
p(y \mid w, \phi_X) = \mathcal{N}(y; \phi_X^T w, \sigma^2 I)
\]

\[
p(f_x \mid y, \phi_X) = \mathcal{N}(f_x; \phi_x^T \mu + \phi_x^T \Sigma \phi_X (\phi_X^T \Sigma \phi_X + \sigma^2 I)^{-1} (y - \phi_X^T \mu), \phi_x^T \Sigma \phi_x - \phi_x^T \Sigma \phi_X (\phi_X^T \Sigma \phi_X + \sigma^2 I)^{-1} \phi_X^T \Sigma \phi_x)
\]
% prior on $w$
\[
F = 2; \quad \text{% number of features}
\]

\[
\text{phi} = @(a)(\text{bsxfun(@power,a,0:F-1))}; \quad \text{% } \phi(a) = [1;a]
\]

\[
\text{mu} = \text{zeros}(F,1); \quad \text{% } p(w) = \mathcal{N}(\mu, \Sigma)
\]

% prior on $f(x)$
\[
\text{n} = 100; \quad x = \text{linspace(-6,6,n)}'; \quad \text{% 'test' points}
\]

\[
\text{phix} = \text{phi}(x); \quad \text{% features of } x
\]

\[
\text{m} = \text{phix} * \text{mu}; \quad \text{% } p(f_x) = \mathcal{N}(m, k_{xx})
\]

\[
\text{kxx} = \text{phix} * \text{Sigma} * \text{phix}'; \quad \text{% } p(f_x) = \mathcal{N}(M, k_{XX})
\]

\[
\text{s} = \text{bsxfun(@plus,m,\text{chol(kxx + 1.0e-8 * eye(n))'} * \text{randn(n,3)))); \quad \text{% samples from prior}
\]

\[
\text{stdpi} = \text{sqrt(diag(kxx))}; \quad \text{% marginal stddev, for plotting}
\]

load('data.mat'); \text{N} = \text{length(Y)}; \quad \text{% gives Y,X,sigma}

% prior on $Y = f_X + \epsilon$
\[
\text{phiX} = \text{phi}(X); \quad \text{% features of data}
\]

\[
\text{M} = \text{phiX} * \text{mu}; \quad \text{% } p(f_X) = \mathcal{N}(M, k_{XX})
\]

\[
\text{kXX} = \text{phiX} * \text{Sigma} * \text{phiX}'; \quad \text{% } p(Y) = \mathcal{N}(M, k_{XX} + \sigma^2 I)
\]

\[
\text{G} = \text{kXX} + \text{sigma}^2 * \text{eye(N)}; \quad \text{% most expensive step: } \mathcal{O}(N^3)
\]

\[
\text{R} = \text{chol(G)};
\]

\[
\text{kxX} = \text{phix} * \text{Sigma} * \text{phiX}'; \quad \text{% cov}(f_x, f_X) = k_{xx}
\]

\[
\text{A} = \text{kxX} / \text{R}; \quad \text{% pre-compute for re-use}
\]

\[
\text{mpost} = \text{m} + \text{A} * (\text{R'} \backslash (\text{Y-M})); \quad \text{% } p(f_x | Y) = \mathcal{N}(m + k_{xx}X(k_{XX} + \sigma^2 I)^{-1}(Y - M), k_{xx} - k_{xx}X(k_{XX} + \sigma^2 I)^{-1}k_{xX})
\]

\[
\text{vpost} = \text{kxx} - \text{A} * \text{A'};
\]

\[
\text{spost} = \text{bsxfun(@plus,mpost,\text{chol(vpost + 1.0e-8 * eye(n))'} * \text{randn(n,3))}); \quad \text{% samples}
\]

\[
\text{stdpo} = \text{sqrt(diag(vpost))}; \quad \text{% marginal stddev, for plotting}
\]
A More Realistic Dataset
General Linear Regression

\[ f(x) = \phi_x^T w \]
\[ f(x) = w_1 + w_2 x = \phi_x^T w \]

\[ \phi_x := \begin{pmatrix} 1 \\ x \end{pmatrix} \]
% prior on $w$
F = 2; % number of features
phi = @(a)(bsxfun(@power,a,0:F-1)); % \( \phi(a) = [1; a] \)
mu = zeros(F,1);
Sigma = eye(F); % \( p(w) = \mathcal{N}(\mu, \Sigma) \)

% prior on $f(x)$
n = 100; x = linspace(-6,6,n)'; % 'test' points
phix = phi(x); % features of $x$
m = phix * mu;
kxx = phix * Sigma * phix'; % \( p(f_x) = \mathcal{N}(m, k_{xx}) \)
s = bsxfun(@plus,m,chol(kxx + 1.0e-8 * eye(n))' * randn(n,3)); % samples from prior
stdpi = sqrt(diag(kxx)); % marginal stddev, for plotting
load('data.mat'); N = length(Y); % gives $Y,X,sigma$

% prior on $Y = f_X + \epsilon$
phiX = phi(X); % features of data
M = phiX * mu;
kXX = phiX * Sigma * phiX'; % \( p(f_X) = \mathcal{N}(M, k_{XX}) \)
G = kXX + sigma^2 * eye(N); % \( p(Y) = \mathcal{N}(M, k_{XX} + \sigma^2 I) \)
R = chol(G); % most expensive step: \( \mathcal{O}(N^3) \)

kxX = phix * Sigma * phiX'; % \( \text{cov}(f_x, f_X) = k_{xx} \)
A = kxX / R; % pre-compute for re-use

mpost = m + A * (R' \ (Y-M)); % \( p(f_x | Y) = \mathcal{N}(m + k_{xx} (k_{XX} + \sigma^2 I)^{-1} (Y - M), \ k_{xx} - k_{xx} (k_{XX} + \sigma^2 I)^{-1} k_{xx} ) \)
vpost = kxx - A * A';
spost = bsxfun(@plus,mpost,chol(vpost + 1.0e-8 * eye(n))' * randn(n,3)); % samples
stdpo = sqrt(diag(vpost)); % marginal stddev, for plotting
Cubic Regression

\[ f(x) = \phi(x)^\top w \quad \phi(x) = \begin{pmatrix} 1 & x & x^2 & x^3 \end{pmatrix}^\top \]
Cubic Regression

\[
f(x) = \phi(x)^\top w \\
\phi(x) = \begin{pmatrix} 1 & x & x^2 & x^3 \end{pmatrix}^\top
\]
Septic Regression?

\[ \phi = @(a)(\text{bsxfun}(\@power,a,[0:7])); \]

\[ f(x) = \phi(x)^T w \quad \phi(x) = \begin{pmatrix} 1 & x & x^2 & \cdots & x^7 \end{pmatrix}^T \]
$f(x) = \phi(x)^\top w$

$\phi(x) = \begin{pmatrix} 1 & x & x^2 & \cdots & x^7 \end{pmatrix}^\top$
Fourier Regression

\( \phi(x) = (\cos(x) \quad \cos(2x) \quad \cos(3x) \quad \ldots \quad \sin(x) \quad \sin(2x) \quad \ldots)^T \)
Fourier Regression

\[ \phi(x) = \left( \cos(x) \cos(2x) \cos(3x) \ldots \sin(x) \sin(2x) \ldots \right)^T \]
Step Regression

\[ \phi(x) = -1 + 2 \left( \theta(x - 8) \quad \theta(8 - x) \quad \theta(x - 7) \quad \theta(7 - x) \quad \ldots \right)^T \]

\[
\phi = @(a)(-1 + 2 * \text{bsxfun}(@lt,a,\text{linspace}(-8,8,16))); 
\]
Step Regression

\[
\phi(x) = -1 + 2 \left( \theta(x - 8) \quad \theta(8 - x) \quad \theta(x - 7) \quad \theta(7 - x) \quad \ldots \right)^T
\]

\[\phi = @(a)(-1 + 2 * bsxfun(@lt,a,linspace(-8,8,16)));\]
Another Kind of Step Regression

$$\phi(x) = \left( \theta(x - 8) \quad \theta(8 - x) \quad \theta(x - 7) \quad \theta(7 - x) \quad \ldots \right)^T$$
Another Kind of Step Regression

\[
\phi(x) = (\theta(x - 8) \quad \theta(8 - x) \quad \theta(x - 7) \quad \theta(7 - x) \quad \ldots)^T
\]
V Regression

\[
\phi(x) = (|x - 8| + 8 \quad |x - 7| + 7 \quad |x - 6| + 6 \quad \ldots)^T
\]

\[
\phi(x) = \begin{pmatrix}
|x - 8| + 8 \\
|x - 7| + 7 \\
|x - 6| + 6 \\
\vdots
\end{pmatrix}
\]
\[ \phi(x) = (|x - 8| + 8 \quad |x - 7| + 7 \quad |x - 6| + 6 \quad \ldots)^T \]
Legendre Regression

\[
\phi(x) = \left( b^0 P_0(x), b^1 P_1(x), \ldots, b^{13} P_{13}(x) \right)^	op \\

P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n
\]
Legendre Regression

\[
\phi(x) = (b^0 P_0(x), b^1 P_1(x), \ldots, b^{13} P_{13}(x))^{\top}
\]

\[
P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n
\]
Eiffel Tower Regression

\[
\phi(x) = \begin{pmatrix} e^{-|x-8|} & e^{-|x-7|} & e^{-|x-6|} & \ldots \end{pmatrix}^\top
\]

\[
\text{phi} = @(x)\left(\exp(-\text{abs}(\text{bsxfun(@minus,a,[\ldots]))})\right);
\]
Eiffel Tower Regression

\[ \phi(x) = (e^{-|x-8|} \quad e^{-|x-7|} \quad e^{-|x-6|} \ldots)^\top \]
Bell Curve Regression

\[
\phi(x) = \begin{pmatrix} e^{-\frac{1}{2}(x-8)^2} & e^{-\frac{1}{2}(x-7)^2} & e^{-\frac{1}{2}(x-6)^2} & \ldots \end{pmatrix}^T
\]
Bell Curve Regression

\[ \phi(x) = \left( e^{-\frac{1}{2}(x-8)^2} \quad e^{-\frac{1}{2}(x-7)^2} \quad e^{-\frac{1}{2}(x-6)^2} \quad \ldots \right)^T \]

\[
\text{phi} = @(a)(exp(-0.5 * bsxfun(@minus,a,[-8:1:8]).^2));
\]
Multiple Inputs

all this works for in multiple dimensions, too

\[ \phi : \mathbb{R}^N \rightarrow \mathbb{R} \quad f : \mathbb{R}^N \rightarrow \mathbb{R} \]
Multiple Inputs

all this works for in multiple dimensions, too
Multiple Outputs

slightly more confusing, but no algebraic problem

\[ \phi : \mathbb{R} \rightarrow \mathbb{R}^M \quad f : \mathbb{R} \rightarrow \mathbb{R}^M \quad \text{cov}(f_i(t), f_j(t)) = \sum_\ell \phi_{\ell,i}(t)\phi_{\ell,j}(t') \]

- \( [f_1(t_1), \ldots, f_1(t_N), f_2(t_1), \ldots, f_2(t_N), \ldots, f_M(t_1), \ldots, f_M(t_N)] \)
  are just some co-varying Gaussian variables
- requires careful matrix algebra
Multiple Outputs

learning paths

\[ \phi : \mathbb{R} \rightarrow \mathbb{R}^M \quad f : \mathbb{R} \rightarrow \mathbb{R}^M \quad \text{cov}(f_i(t), f_j(t)) = \sum_\ell \phi_{\ell,i}(t)\phi_{\ell,j}(t') \]

- \([f_1(t_1), \ldots, f_1(t_N), f_2(t_1), \ldots, f_2(t_N), \ldots, f_M(t_1), \ldots, f_M(t_N)]\)
  - are just some co-varying Gaussian variables
- requires careful matrix algebra
Multiple Outputs
learning paths

\[ \phi : \mathbb{R} \to \mathbb{R}^M \quad f : \mathbb{R} \to \mathbb{R}^M \quad \text{cov}(f_i(t), f_j(t)) = \sum_{\ell} \phi_{\ell,i}(t) \phi_{\ell,j}(t') \]

- \([f_1(t_1), \ldots, f_1(t_N), f_2(t_1), \ldots, f_2(t_N), \ldots, f_M(t_1), \ldots f_M(t_N)]\) are just some co-varying Gaussian variables
- requires careful matrix algebra
How many features should we use?

let's look at that algebra again

\[
p(f_x | y, \phi_X) = \mathcal{N}(f_x; \phi_x^T \mu + \phi_x^T \Sigma \phi_X (\phi_X^T \Sigma \phi_X + \sigma^2 I)^{-1}(y - \phi_X^T \mu), \\
\phi_x^T \Sigma \phi_x - \phi_x^T \Sigma \phi_X (\phi_X^T \Sigma \phi_X + \sigma^2 I)^{-1} \phi_X^T \Sigma \phi_x)
\]

- there's no lonely \( \phi \) in there
- all objects involving \( \phi \) are of the form
  - \( \phi^T \mu \) — the mean function
  - \( \phi^T \Sigma \phi \) — the kernel
- once these are known, cost is independent of the number of features
- remember the code:
  ```matlab
  M = phiX * mu;
  m = phix * mu;
  kXX = phiX * Sigma * phiX'; % p(f_X) = \mathcal{N}(M, k_{X X})
  kxx = phix * Sigma * phix'; % p(f_x) = \mathcal{N}(m, k_{xx})
  kxX = phix * Sigma * phiX'; % \text{cov}(f_x, f_X) = k_{x X}
  ```
% prior on \( w \)
\[
F = 2; \quad % \text{number of features} \\
\phi = @(a)(bsxfun(@power,a,0:F-1)); \quad % \phi(a) = [1;a] \\
mu = zeros(F,1); \\
Sigma = eye(F); \quad % p(w) = \mathcal{N}(\mu,\Sigma)
\]

% prior on \( f(x) \)
\[
n = 100; \quad x = linspace(-6,6,n)'; \quad % \text{‘test’ points} \\
phix = phi(x); \\
m = phix * mu; \\
kxx = phix * Sigma * phix'; \quad % p(f_x) = \mathcal{N}(m,k_{xx}) \\
s = bsxfun(@plus,m,chol(kxx + 1.0e-8 * eye(n))' * randn(n,3)); \quad % \text{samples from prior} \\
stdpi = sqrt(diag(kxx)); \quad % \text{marginal stddev, for plotting}
\]

load('data.mat'); \( N = \text{length}(Y); \quad % \text{gives Y,X,sigma}
\]

% prior on \( Y = f_X + \epsilon \)
\[
\text{phiX} = phi(X); \quad % \text{features of data} \\
M = phiX * mu; \\
kXX = phiX * Sigma * phiX'; \quad % p(f_X) = \mathcal{N}(M,k_{XX}) \\
G = kXX + sigma^2 * eye(N); \quad % p(Y) = \mathcal{N}(M,k_{XX} + \sigma^2 I) \\
R = chol(G); \quad % \text{most expensive step: } \mathcal{O}(N^3) \\
kXX = phiX * Sigma * phiX'; \\
A = kXX / R; \quad % \text{cov}(f_x,f_X) = k_{xx} \\
\]
\[
\text{mpost} = m + A * (R' \backslash (Y-M)); \quad % p(f_x | Y) = \mathcal{N}(m + k_{xx} X (k_{XX} + \sigma^2 I)^{-1}(Y - M), k_{xx} - k_{xx} X (k_{XX} + \sigma^2 I)^{-1} k_{xx}) \\
\text{vpost} = kXX - A * A'; \\
\text{spost} = bsxfun(@plus,mpost,chol(vpost + 1.0e-8 * eye(n))' * randn(n,3)); \quad % \text{samples} \\
\text{stdpo} = sqrt(diag(vpost)); \quad % \text{marginal stddev, for plotting}
\]
% prior
F = 2; % number of features
phi = @(a)(bsxfun(@power,a,0:F)); % φ(a) = [1;a]
k = @(a,b)(phi(a)' * phi(b)); % kernel
mu = @(a)(zeros(size(a,1))); % mean function

% belief on \( f(x) \)
\( n = 100; x = \text{linspace}(-6,6,n)' \); % ‘test’ points
m = mu(x);
kxx = k(x,x); % \( p(f_x) = \mathcal{N}(m,k_{xx}) \)
s = bsxfun(@plus,m,chol(kxx + 1.0e-8 * eye(n))' * randn(n,3)); % samples from prior
stdpi = sqrt(diag(kxx)); % marginal stddev, for plotting

load('data.mat'); N = length(Y); % gives Y,X,sigma

% prior on \( Y = f_X + \epsilon \)
M = mu(X);
kXX = k(X,X); % \( p(f_X) = \mathcal{N}(M,k_{XX}) \)
G = kXX + sigma^2 * eye(N); % \( p(Y) = \mathcal{N}(M,k_{XX} + \sigma^2 I) \)
R = chol(G); % most expensive step: \( \mathcal{O}(N^3) \)
kXX = k(x,X);
A = kXX / R; % pre-compute for re-use

mpost = m + A * (R' \ (Y-M)); % \( p(f_X | Y) = \mathcal{N}(m + k_{XX} \ (k_{XX} + \sigma^2 I)^{-1} (Y - M), k_{XX} - k_{XX} \ (k_{XX} + \sigma^2 I)^{-1} k_{XX}) \)
vpost = kxx - A * A';
spost = bsxfun(@plus,mpost,chol(vpost + 1.0e-8 * eye(n))' * randn(n,3)); % samples
stdpo = sqrt(diag(vpost)); % marginal stddev, for plotting
Features are cheap, so let’s use a lot

For simplicity, let’s fix \( \Sigma = \frac{\sigma^2 (c_{\text{max}} - c_{\text{min}})}{F} I \).

The elements of \( \phi_x^T \Sigma \phi_x \) are

\[
\phi(x_i)^T \Sigma \phi(x_j) = \frac{\sigma^2 (c_{\text{max}} - c_{\text{min}})}{F} \sum_{\ell=1}^{F} \phi(x_i) \phi(x_j)
\]

\( \text{phi}=@(a)(\exp(-0.5 * \text{bsxfun(@minus,a,[-8:1:8]).^2})./s.^2); \)

\[
\phi(x) = \exp \left( -\frac{(x - c_\ell)^2}{2\lambda^2} \right)
\]

\[
\phi(x_i)^T \Sigma \phi(x_j) = \frac{\sigma^2 (c_{\text{max}} - c_{\text{min}})}{F} \sum_{\ell=1}^{F} \exp \left( -\frac{(x_i - c_\ell)^2}{2\lambda^2} \right) \exp \left( -\frac{(x_j - c_\ell)^2}{2\lambda^2} \right)
\]

\[
= \frac{\sigma^2 (c_{\text{max}} - c_{\text{min}})}{F} \exp \left( -\frac{(x_i - x_j)^2}{4\lambda^2} \right) \sum_{\ell} \exp \left( -\frac{(c_\ell - \frac{1}{2}(x_i + x_j))^2}{\lambda^2} \right)
\]
Features are cheap, so let’s use a lot of them.

\[ \phi(x_i)^\top \Sigma \phi(x_j) = \sigma^2 \frac{(c_{\text{max}} - c_{\text{min}})}{F} \exp \left( -\frac{(x_i - x_j)^2}{4\lambda^2} \right) \sum_{\ell} \exp \left( -\frac{(c_{\ell} - \frac{1}{2}(x_i + x_j))^2}{\lambda^2} \right) \]

- now increase \( F \), such that # of features in \( \delta c \) becomes

\[ \frac{F \cdot \delta c}{(c_{\text{max}} - c_{\text{min}})} \]

\[ \phi(x_i)^\top \Sigma \phi(x_j) \rightarrow \sigma^2 \exp \left( -\frac{(x_i - x_j)^2}{4\lambda^2} \right) \int_{c_{\text{min}}}^{c_{\text{max}}} \exp \left( -\frac{(c - \frac{1}{2}(x_i + x_j))^2}{\lambda^2} \right) \, dc \]

- let \( c_{\text{min}} \rightarrow -\infty, c_{\text{max}} \rightarrow \infty \)

\[ \phi(x_i)^\top \Sigma \phi(x_j) \rightarrow \sqrt{2\pi} \lambda \sigma^2 \exp \left( -\frac{(x_i - x_j)^2}{4\lambda^2} \right) \]
Exponentiated Squares

\[
\phi = @(a) (\exp(-0.5 \times \text{bsxfun(@minus,a,linspace(-8,8,10))).^2 ./ \text{ell}.^2));
\]
Exponentiated Squares

\[ \phi = @(a)(\exp(-0.5 * \text{bsxfun}(@\text{minus}, a, \text{linspace(-8, 8, 30))).^2 ./ \text{ell}.^2)); \]
Exponentiated Squares

\[ k = @(a,b)(5*\exp(-0.25*\text{bsxfun(@minus,a,b').}^2)); \]

\textit{aka. radial basis function, square(d)-exponential kernel}
Exponentiated Squares

\[ k = @(a,b)(5*\exp(-0.25*\text{bsxfun(@minus,a,b').}^2)); \]

- aka. radial basis function, square(d)-exponential kernel
Definition

A function $k : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$ is a **Mercer kernel** if, for any finite collection $X = [x_1, \ldots, x_N]$, the matrix $k_{XX} \in \mathbb{R}^{N \times N}$ with elements $k_{XX}(i,j) = k(x_i, x_j)$ is **positive semidefinite**.

Lemma

Any kernel that can be written as

$$k(x, x') = \int \phi_{\ell}(x) \phi_{\ell}(x') \, d\ell$$

is a Mercer kernel. (assuming integral over positive set)

**Proof:** $\forall X \in \mathbb{X}^N, v \in \mathbb{R}^N$

$$v^\top k_{XX} v = \int \sum_i v_i \phi_{\ell}(x_i) \sum_j v_j \phi_{\ell}(x_j) \, d\ell = \int \left[ \sum_i v_i \phi_{\ell}(x_i) \right]^2 \, d\ell \geq 0 \quad \square$$
Definition

A function \( k : X \times X \to \mathbb{R} \) is a **Mercer kernel** if, for any finite collection \( X = [x_1, \ldots, x_N] \), the matrix \( k_{XX} \in \mathbb{R}^{N\times N} \) with elements \( k_{XX},(i,j) = k(x_i, x_j) \) is positive semidefinite.

Definition

Let \( \mu : X \to \mathbb{R} \) be any function, \( k : X \times X \to \mathbb{R} \) be a Mercer kernel. A **Gaussian process** \( p(f) = \mathcal{GP}(f; \mu, k) \) is a probability distribution over the function \( f : X \to \mathbb{R} \), such that every finite restriction to function values \( f_X := [f_{x_1}, \ldots, f_{x_N}] \) is a **Gaussian distribution** \( p(f_X) = \mathcal{N}(f_X; \mu_X, k_{XX}) \).
Those step functions

\[
\phi = @(a)(\text{bsxfun}(\text{gt}, a, \text{linspace}(-8, 8, 5))./\text{sqrt}(5));
\]
Those step functions

\[ \phi = @(a)(\text{bsxfun}(\text{gt}, a, \text{linspace}(-8, 8, 20))./\sqrt{20}) \]
Those step functions

\[
\phi = @(a) \left( \frac{\text{bsxfun}(@gt, a, \text{linspace}(-8, 8, 100))}{\sqrt{100}} \right);
\]
Those step functions

\[ k = @(a,b)(\theta^2 \times \text{bsxfun(@min,a+8,b+8)/16}); \]

\[ \text{cov}(f_{x_i}, f_{x_j}) = \int_{c_{\text{min}}}^{\infty} \theta(x_i - c)\theta(x_j - c) \, dc = \min(x_i, x_j) - c_{\text{min}} \]

- aka. the Wiener process
Those step functions

\[ k = @(a,b)(\theta^2 \times \text{bsxfun(@min,a+8,b'+8)/16}); \]
Those other step-functions

\[
\phi = @(a)(-1 + 2 \times \text{bsxfun}(@lt,a,linspace(-8,8,5)));
\]

Wahba, 1990
Those other step-functions

\[
\phi = @(a)\left(-1 + 2 \times \text{bsxfun}(@lt,a,\text{linspace}(-8,8,20))\right);
\]
Those other step-functions

\[
\phi = @(a)(-1 + 2 \times \text{bsxfun}(@lt,a,\text{linspace}(-8,8,100)));
\]

Wahba, 1990
Those other step-functions

\[
k = @(a,b)((1 + c - 2 * c * \text{abs}(\text{bsxfun(@minus,a,b')}/16)));
\]

Wahba, 1990

\[
\text{cov}(f_{x_i}, f_{x_j}) = 1 + b \int_{0}^{1} (2\theta(x_i-c) - 1)(2\theta(x_j-c) - 1) \, dc = 1 + b - 2b|x_i - x_j|
\]

- aka. linear splines
Those other step-functions

\[ k = @(a,b)((1 + c - 2 * c * \text{abs}(\text{bsxfun}(@\text{minus},a,b')/16))); \]

\[ \text{cov}(f_{x_i}, f_{x_j}) = 1 + b \int_0^1 (2\theta(x_i - c) - 1)(2\theta(x_j - c) - 1) \, dc = 1 + b - 2b|x_i - x_j| \]

\vspace{0.5cm}
\textit{aka.} linear splines
Those linear features

\[
\phi = @(a)(\text{bsxfun}(@minus,\text{abs}(\text{bsxfun}(@minus,a,\text{linspace}(-8,8,5))))) \text{linspace}(-8,8,5));
\]

Wahba, 1990

\[
\cov(f_i, f_j) = 1 + x_i x_j + \frac{b}{\int_0^1 (x_i - c - c)(x_j - c - c) dc} = 1 + \left(1 + \frac{b}{3}\right)x_i x_j + \frac{b}{3}(x_i - x_j)/3 - x_3 - y_3
\]
aka. cubic splines
Those linear features

\[
\phi = @(a) (\text{bsxfun}(@minus, \text{abs}(\text{bsxfun}(@minus, a, \text{linspace}(-8, 8, 20))), \text{linspace}(-8, 8, 20)));
\]

Wahba, 1990

\[
\text{cov}(f_x^i, f_x^j) = 1 + x^i x^j + \frac{b}{\text{integral} \cdot \text{disp}} \int_0^1 \left( x^i - c \right)^\frac{3}{\text{disp}} - \left( x^j - c \right)^\frac{3}{\text{disp}} \text{d}c = 1 + \left( 1 + \frac{b}{3} \right) x^i x^j + \frac{b}{3} \left( x^i - x^j \right)^\frac{3}{3} - \left( x^3 - y^3 \right)
\]
aka. cubic splines
Those linear features

\[
\text{Wahba, 1990}
\]

\[
\phi = @(a)(\text{bsxfun}(\oplus, \text{abs}(\text{bsxfun}(\ominus, a, \text{linspace}(-8, 8, 100))), \text{linspace}(-8, 8, 100)));
\]

\[
\text{cov}(f_i, f_j) = 1 + x_i x_j + \frac{b}{\text{integral}} \frac{1}{x_i - c - c} - \frac{1}{x_j - c - c} = 1 + \left(1 + \frac{b}{3}\right)x_i x_j + \frac{b}{3} (x_i - x_j)
\]

aka. cubic splines
Those linear features

$$k = @(a,b)(\theta^2 \ast (1 + (1+c) \ast \text{bsxfun}(@times,a+8,b'+8)/16 + c / 3 \ast \text{abs(bsxfun}(@minus,a,b')/16)^3 - \text{bsxfun}(@plus,((a+8)/16)^3,((b'+8)/16)^3))))$$

$$\text{cov}(f_{x_i}, f_{x_j}) = 1 + x_i x_j + b \int_0^1 (|x_i - c| - c)(|x_j - c| - c) \, dc$$

$$= 1 + (1 + b)x_i x_j + \frac{b}{3}(|x_i - x_j|^3 - x^3 - y^3)$$

aka. cubic splines
Those linear features

\[
k = @(a,b)(\theta^2 \ast (1 + (1+c) \ast \text{bsxfun}(@times,a+8,b'+8)/16 + c/3 \ast (\text{abs}(	ext{bsxfun}(@minus,a,b'))/16)^3 - \text{bsxfun}(@plus,((a+8)/16)^3,((b'+8)/16)^3))));
\]

Wahba, 1990

\[
\text{cov}(f_{x_i}, f_{x_j}) = 1 + x_i x_j + b \int_0^1 (|x_i - c| - c)(|x_j - c| - c) \, dc
\]

\[
= 1 + (1 + b)x_i x_j + \frac{b}{3}(|x_i - x_j|^3 - x^3 - y^3)
\]

aka. cubic splines
Exponentially suppressed polynomials

\[ \phi(a) = \text{bsxfun}(\times, \text{bsxfun}(\text{power}, a./9, [0:1]), c.^{[0:1]}) \]

\[ \text{cov}(f_{x_i}, f_{x_j}) = \sum_{\ell=0}^{1} b^\ell x_i^\ell x_j^\ell \quad 0 \leq b \leq 1 \quad -1 < x_i, x_j < 1 \]
Exponentially suppressed polynomials

\[
\phi = @(a)(\text{bsxfun}(\times, \text{bsxfun}(\text{power}, a./9, [0:2]), c.^[0:2]));
\]

\[
\text{cov}(f_{x_i}, f_{x_j}) = \sum_{\ell=0}^{2} b^\ell x_i^\ell x_j^\ell \quad 0 \leq b \leq 1 \quad -1 < x_i, x_j < 1
\]
Exponentially suppressed polynomials

\[
\phi(a) = \text{bsxfun}(\times, \text{bsxfun}(\text{power}, a ./ 9, [0:10]), c.^{[0:10]});
\]

\[
\text{cov}(f_{x_i}, f_{x_j}) = \sum_{\ell=0}^{10} b^\ell x_i^\ell x_j^\ell \quad 0 \leq b \leq 1 \quad -1 < x_i, x_j < 1
\]

Minka, 2000
Exponentially suppressed polynomials

\[ k = @(a,b)(\theta^2 \cdot \frac{1}{1-c*bsxfun(@times,a./8,b'./8)}) \]

\[
\text{cov}(f_{x_i}, f_{x_j}) = \sum_{\ell=0}^{\infty} b^{\ell} x_i^{\ell} x_j^{\ell} = \frac{1}{1 - bx_i x_j} \quad 0 \leq b \leq 1 \quad -1 < x_i, x_j < 1
\]
Exponentially suppressed polynomials

\[
k = @(a,b)(\theta^2 \cdot \frac{1}{1 - c \cdot \text{bsxfun}(\times, a./8, b'./8)})
\]

\[
\operatorname{cov}(f_{x_i}, f_{x_j}) = \sum_{\ell=0}^{\infty} b^\ell x_i^\ell x_j^\ell = \frac{1}{1 - bx_i x_j}
\]

\[0 \leq b \leq 1 \quad -1 < x_i, x_j < 1\]
Exponentially decaying periodic features

\[
\phi(a) = \text{bsxfun}(@times, \cos(\text{bsxfun}(@times, a/8, [0:2])), c.^[0:2]), \ldots \text{bsxfun}(@times, \sin(\text{bsxfun}(@times, a/8, [1:2])), c.^[1:2]));
\]

\[
\text{cov}(f_{x_i}, f_{x_j}) = 1 + \sum_{\ell=0}^{2} b^\ell (\cos(2\pi \ell x_i) \cos(2\pi \ell x_j) + \sin(2\pi \ell x_i) \sin(2\pi \ell x_j))
\]

\[0 \leq b \leq 1\]
Exponentially decaying periodic features

\[
\phi = @(a)(\text{bsxfun}(\text{@times},\cos(\text{bsxfun}(\text{@times},a/8,[0:20])),c.^[0:20]), \ldots \text{bsxfun}(\text{@times},\sin(\text{bsxfun}(\text{@times},a/8,[1:20])),c.^[1:20]));
\]

\[
\text{cov}(f_{x_i}, f_{x_j}) = 1 + \sum_{\ell=0}^{20} b^\ell (\cos(2\pi \ell x_i) \cos(2\pi \ell x_j) + \sin(2\pi \ell x_i) \sin(2\pi \ell x_j))
\]

\[
0 \leq b \leq 1
\]
Exponentially decaying periodic features

\[
\phi = @(a)(\text{bsxfun}(@times,\cos(\text{bsxfun}(@times,a/8,[0:50])),c.^{[0:50]}), \ldots \\
\text{bsxfun}(@times,\sin(\text{bsxfun}(@times,a/8,[1:50])),c.^{[1:50]}));
\]

\[
\text{cov}(f_{x_i}, f_{x_j}) = 1 + \sum_{\ell=0}^{50} b^\ell (\cos(2\pi \ell x_i) \cos(2\pi \ell x_j) + \sin(2\pi \ell x_i) \sin(2\pi \ell x_j) )
\]

\[0 \leq b \leq 1\]
Exponentially decaying periodic features

\[ k = @(a,b)(\theta^2 \cdot \frac{2}{\pi} \cdot \frac{\sin(2 \cdot (\text{cr} + \text{cu} \cdot \text{bsxfun(@times,a,b'))})}{\sqrt{\text{bsxfun(@times,(1 + 2 \cdot (\text{cr} + \text{cu} \cdot a.^2)),(1 + 2 \cdot (\text{cr} + \text{cu} \cdot b'.^2))))}}) \];

\[
\text{cov}(f_{x_i}, f_{x_j}) = 1 + \sum_{\ell=0}^{\infty} b^\ell (\cos(2\pi \ell x_i) \cos(2\pi \ell x_j) + \sin(2\pi \ell x_i) \sin(2\pi \ell x_j))
\]

\[
= \frac{1}{2} + \frac{(1 - b^2)/2}{1 + b^2 - 2b \cos(2\pi (x_i - x_j))} \quad 0 \leq b \leq 1
\]
Exponentially decaying periodic features

\[ k(a,b) = \theta^2 \cdot \frac{2}{\pi} \cdot \frac{\sin(\theta)}{\sqrt{(1 + 2 \cdot (\cos(a^2) + \cos(b^2)))(1 + 2 \cdot (\cos(a^2) + \cos(b^2))})}; \]

\[
\text{cov}(f_{x_i}, f_{x_j}) = 1 + \sum_{\ell=0}^{\infty} b^\ell \left( \cos(2\pi \ell x_i) \cos(2\pi \ell x_j) + \sin(2\pi \ell x_i) \sin(2\pi \ell x_j) \right) \\
= \frac{1}{2} + \frac{(1 - b^2)/2}{1 + b^2 - 2b \cos(2\pi (x_i - x_j))} \quad 0 \leq b \leq 1.
\]
“White Noise”
the “limit” of block functions

\[
\lim_{\epsilon \to 0} \int \mathbb{I}(|x_i - c| < \epsilon) \mathbb{I}(|x_j - c| < \epsilon) \, dc = \delta(x_i - x_j)
\]

- but we’re cheating a little (height of blocks goes to 0!)
- white noise is a concept, more than a proper limit
- if you make no assumptions, you learn nothing
“White Noise”

the “limit” of block functions

\[
\lim_{\epsilon \to 0} \int \mathbb{I}(|x_i - c| < \epsilon) \mathbb{I}(|x_j - c| < \epsilon) \, dc = \delta(x_i - x_j)
\]

- but we’re cheating a little (height of blocks goes to 0!)
- white noise is a concept, more than a proper limit
- if you make no assumptions, you learn nothing
“White Noise”
the “limit” of block functions

\[ \lim_{\epsilon \to 0} \int \mathbb{I}(|x_i - c| < \epsilon) \mathbb{I}(|x_j - c| < \epsilon) \, dc = \delta(x_i - x_j) \]

- but we’re cheating a little (height of blocks goes to 0!)
- white noise is a concept, more than a proper limit
- if you make no assumptions, you learn nothing
“White Noise”
the “limit” of block functions

\[
\lim_{\epsilon \to 0} \int \mathbb{I}(|x_i - c| < \epsilon) \mathbb{I}(|x_j - c| < \epsilon) \, dc = \delta(x_i - x_j)
\]

- but we’re cheating a little (height of blocks goes to 0!)
- white noise is a concept, more than a proper limit
- if you make no assumptions, you learn nothing
“White Noise”
the “limit” of block functions

\[ \lim_{\epsilon \to 0} \int \mathbb{I}(\lvert x_i - c \rvert < \epsilon) \mathbb{I}(\lvert x_j - c \rvert < \epsilon) \, dc = \delta(x_i - x_j) \]

- but we’re cheating a little (height of blocks goes to 0!)
- white noise is a concept, more than a proper limit
- if you make no assumptions, you learn nothing
That gcd kernel

\[ k = @(a,b)(\gcd(\text{bsxfun(@times,a,ones(size(b')))),\text{bsxfun(@times,ones(size(a)),b'))}); \]
That gcd kernel

\[ k = @(a,b)(\text{gcd(bsxfun(@times,a,ones(size(b')))),bsxfun(@times,ones(size(a)),b'))}; \]
That gcd kernel

\[ k = @(a,b) \left( \gcd( \text{bsxfun(@times, a, ones(size(b')))), \text{bsxfun(@times, ones(size(a)), b'))} \right); \]
Gaussians are closed under
  - linear projection / marginalization / sum rule
  - linear restriction / conditioning / product rule
⇒ they provide the linear algebra of inference
  - combine with nonlinear features $\phi$, get nonlinear regression
  - in fact, number of features can be infinite
⇒ (nonparametric) Gaussian process regression

Tomorrow:
  - so what are kernels? What is the set of kernels?
  - how should we design GP models?
  - how powerful are those models?
Bibliography

- **D.J.C. MacKay**
  *Introduction to Gaussian Processes*
  In Bishop, C.M. (ed.), *Neural Networks and Machine Learning*, Springer, 1998

- **C.E. Rasmussen & C.K.I. Williams**
  *Gaussian Processes for Machine Learning*
  MIT Press, 2006

- **T. Minka**
  *Deriving quadrature rules from Gaussian processes*
  Tech. Report 2000

- **G. Wahba**
  *Spline Models for Observational Data*
  SIAM CBMS-NSF reg. conf. series in applied mathematics, 1990