Gaussian Processes
Introduction II

Philipp Hennig

GPWS 2014
13 Jan 2014

MAX-PLANCK-GESELLSCHAFT

Research Group Elementary Intelligence
Department of Empirical Inference
Max Planck Institute for Intelligent Systems
Tübingen, Germany

Why?

What is it with this man?

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

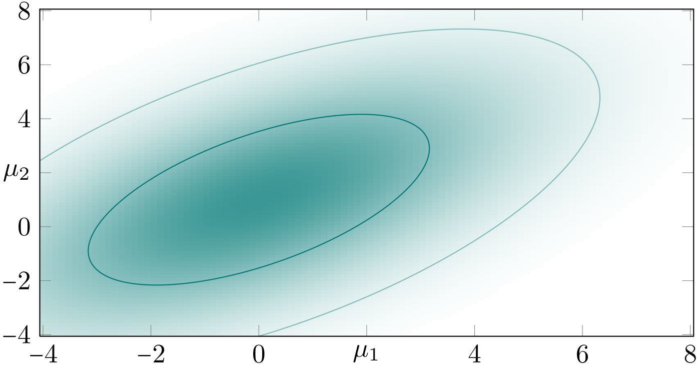
- Gaussians link *inference* and *linear algebra*

Closure Under Multiplication

multiple Gaussian factors form a Gaussian

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

$$C := (A^{-1} + B^{-1})^{-1} \qquad c := C(A^{-1}a + B^{-1}b)$$

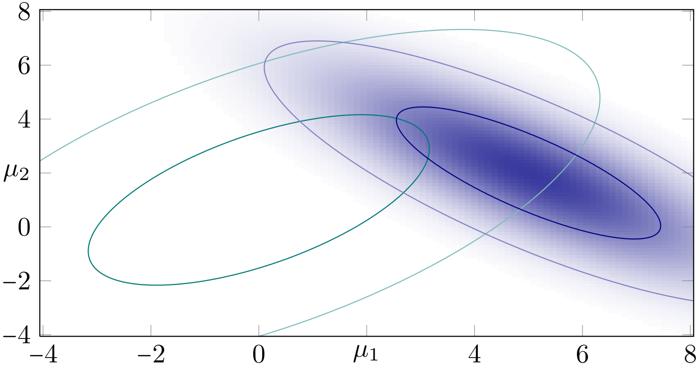


Closure Under Multiplication

multiple Gaussian factors form a Gaussian

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

$$C := (A^{-1} + B^{-1})^{-1} \qquad c := C(A^{-1}a + B^{-1}b)$$

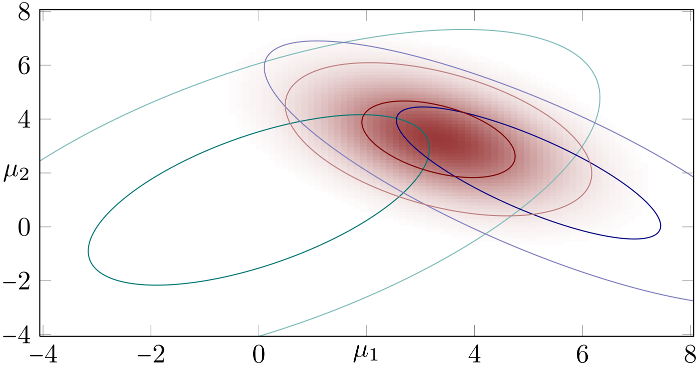


Closure Under Multiplication

multiple Gaussian factors form a Gaussian

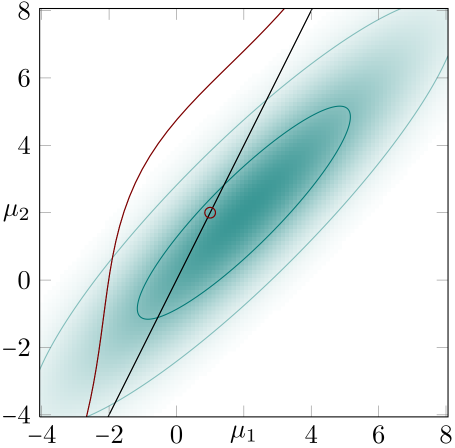
$$\mathcal{N}(x; a, A)\mathcal{N}(x; b, B) = \mathcal{N}(x; c, C)\mathcal{N}(a; b, A + B)$$

$$C := (A^{-1} + B^{-1})^{-1} \qquad c := C(A^{-1}a + B^{-1}b)$$



Closure under Linear Maps

Linear Maps of Gaussians are Gaussians



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

$$\Rightarrow \quad p(Az) = \mathcal{N}(Az, A\mu, A\Sigma A^\top)$$

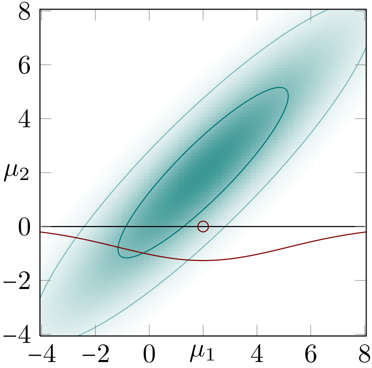
Here: $A = [1, -0.5]$

Closure under Marginalization

projections of Gaussians are Gaussian

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

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



- this is the sum rule

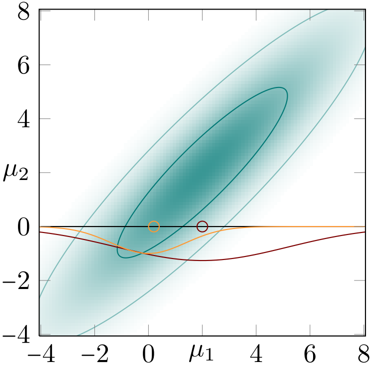
$$\int p(x, y)\,\mathrm{d}y = \int p(y \mid x) p(x)\,\mathrm{d}y = p(x)$$

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

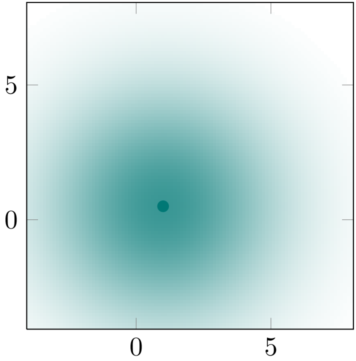
Closure under Conditioning

cuts through Gaussians are Gaussians

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



- this is the product rule
- so Gaussians are closed under the rules of probability



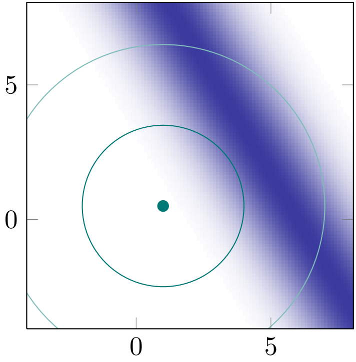
$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)$$

$$= \mathcal{N}\left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 3^2 & 0 \\ 0 & 3^2 \end{pmatrix}\right]$$

$$p\begin{pmatrix} \boldsymbol{x} \\ y \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu} \\ A^\top \boldsymbol{\mu} \end{pmatrix}, \begin{pmatrix} \Sigma & \Sigma A \\ A^\top \Sigma & A^\top \Sigma A + \sigma^2 \end{pmatrix}\right)$$

Bayesian Inference

explaining away



$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)$$

$$= \mathcal{N}\left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 3^2 & 0 \\ 0 & 3^2 \end{pmatrix}\right]$$

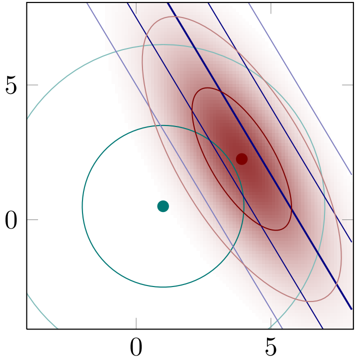
$$p\begin{pmatrix} \boldsymbol{x} \\ y \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu} \\ A^\top \boldsymbol{\mu} \end{pmatrix}, \begin{pmatrix} \Sigma & \Sigma A \\ A^\top \Sigma & A^\top \Sigma A + \sigma^2 \end{pmatrix}\right)$$

$$p(y \mid \boldsymbol{x}, \sigma) = \mathcal{N}(y; A^\top x; \sigma^2)$$

$$= \mathcal{N}\left[6; \begin{pmatrix} 1 & 0.6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \sigma^2\right]$$

Bayesian Inference

explaining away



$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)$$

$$= \mathcal{N}\left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 3^2 & 0 \\ 0 & 3^2 \end{pmatrix}\right]$$

$$p\begin{pmatrix} \boldsymbol{x} \\ y \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu} \\ A^\intercal \boldsymbol{\mu} \end{pmatrix}, \begin{pmatrix} \Sigma & \Sigma A \\ A^\intercal \Sigma & A^\intercal \Sigma A + \sigma^2 \end{pmatrix}\right)$$

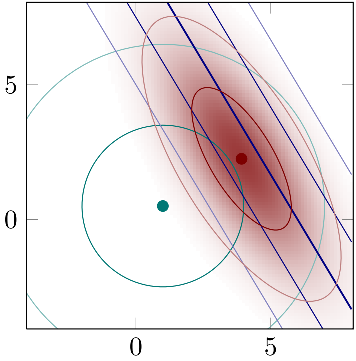
$$p(y \mid \boldsymbol{x}, \sigma) = \mathcal{N}(y; A^\intercal x; \sigma^2)$$

$$= \mathcal{N}\left[6; \begin{pmatrix} 1 & 0.6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \sigma^2\right]$$

$$p(\boldsymbol{x} \mid \sigma^2, y) = \frac{p(\boldsymbol{x}) p(y \mid \boldsymbol{x})}{p(\boldsymbol{y})}$$

Bayesian Inference

explaining away



$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)$$

$$= \mathcal{N}\left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 3^2 & 0 \\ 0 & 3^2 \end{pmatrix}\right]$$

$$p\begin{pmatrix} \boldsymbol{x} \\ y \end{pmatrix} = \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu} \\ A^\top \boldsymbol{\mu} \end{pmatrix}, \begin{pmatrix} \Sigma & \Sigma A \\ A^\top \Sigma & A^\top \Sigma A + \sigma^2 \end{pmatrix}\right)$$

$$p(y \mid \boldsymbol{x}, \sigma) = \mathcal{N}(y; A^\top x; \sigma^2)$$

$$= \mathcal{N}\left[6; \begin{pmatrix} 1 & 0.6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \sigma^2\right]$$

$$p(\boldsymbol{x} \mid \sigma^2, y) = \frac{p(\boldsymbol{x}) p(y \mid \boldsymbol{x})}{p(\boldsymbol{y})}$$

$$= \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu} + \Sigma A (A^\top \Sigma A + \sigma^2)^{-1}(y - A^\top \mu), \Sigma - \Sigma A (A^\top \Sigma A + \sigma^2)^{-1} A^\top \Sigma)$$

$$= \mathcal{N}\left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}; \begin{pmatrix} 3.9 \\ 2.3 \end{pmatrix}, \begin{pmatrix} 3.4 & -3.4 \\ -3.4 & 7.0 \end{pmatrix}\right]$$

Gaussians provide the linear algebra of inference

- products of Gaussians are Gaussians

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

$$C := (A^{-1}+B^{-1})^{-1} \qquad c := C(A^{-1}a + B^{-1}b)$$

- marginals of Gaussians are Gaussians

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

- (linear) conditionals of Gaussians are Gaussians

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

- linear projections of Gaussians are Gaussians

$$p(z) = \mathcal{N}(z;\mu,\Sigma) \quad \Rightarrow \quad p(Az) = \mathcal{N}(Az, A\mu, A\Sigma A^\top)$$

→ Bayesian inference under linear operations

$$p(x) = \mathcal{N}(x;\mu,\Sigma) \qquad p(y \mid x) = \mathcal{N}(y; A^\top x + b, \Lambda)$$

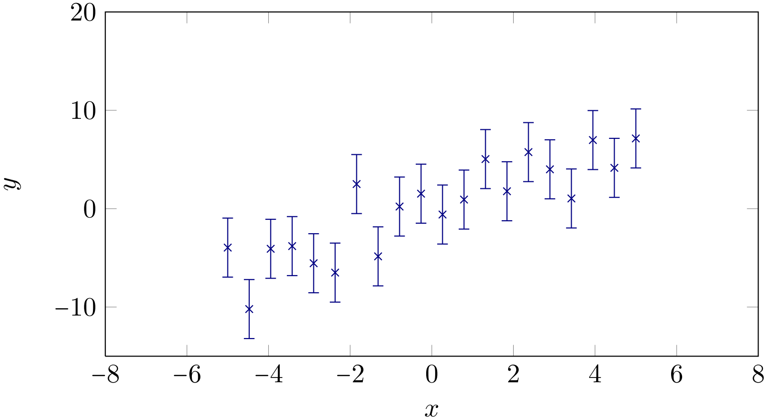
$$p(B^\top x + c \mid y) = \mathcal{N}[B^\top x + c; B^\top \mu + c + B^\top \Sigma A(A^\top \Sigma A + \Lambda)^{-1}(y - A^\top \mu - b),$$
$$B^\top \Sigma B - B^\top \Sigma A(A^\top \Sigma A + \Lambda)^{-1} A^\top \Sigma B]$$

- Gaussians link *inference* and *linear algebra*
- linear weights with features model *functions*

A dataset

linear regression

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

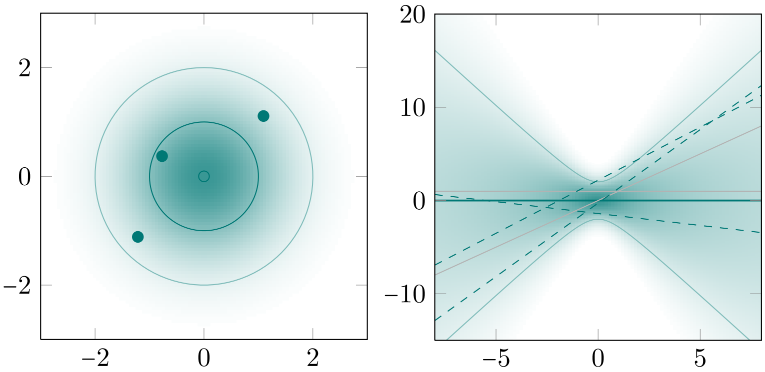


A prior

over linear functions

$$f(x) = w_1 + w_2 x = \phi_x^\top w \qquad \phi_x = \begin{pmatrix} 1 \\ x \end{pmatrix}$$

$$p(w) = \mathcal{N}(w; \mu, \Sigma) \qquad p(f) = \mathcal{N}(f; \phi_x^\top \mu, \phi_x^\top \Sigma \phi_x)$$

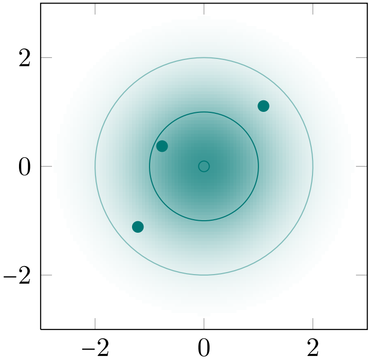


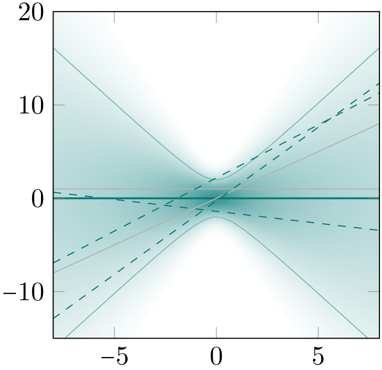
A prior

over linear functions

$$f(x) = w_1 + w_2 x = \phi_x^\top w \qquad \phi_x = \begin{pmatrix} 1 \\ x \end{pmatrix}$$

$$p(w) = \mathcal{N}(w; \mu, \Sigma) \qquad p(f) = \mathcal{N}(f; \phi_x^\top \mu, \phi_x^\top \Sigma \phi_x)$$



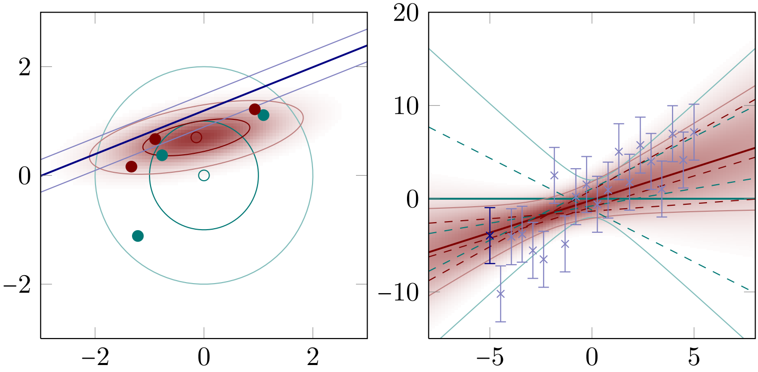


The posterior

over weights

$$p(y \mid w, \phi_X) = \mathcal{N}(y; \phi_X^\top w, \sigma^2 I)$$

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

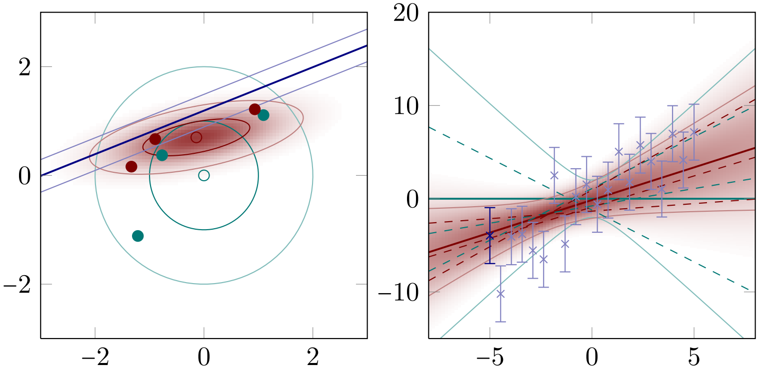


The posterior

over functions

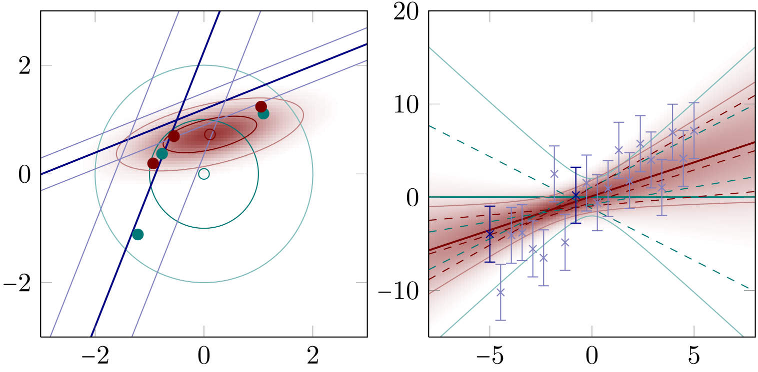
$$p(y \mid w, \phi_X) = \mathcal{N}(y; \phi_X^\top w, \sigma^2 I)$$

$$p(f_x \mid y, \phi_X) = \mathcal{N}(f_x; \phi_x^\top \mu + \phi_x^\top \Sigma \phi_X (\phi_X^\top \Sigma \phi_X + \sigma^2 I)^{-1} (y - \phi_X^\top \mu),$$
$$\phi_x^\top \Sigma \phi_x - \phi_x^\top \Sigma \phi_X (\phi_X^\top \Sigma \phi_X + \sigma^2 I)^{-1} \phi_X^\top \Sigma \phi_x)$$



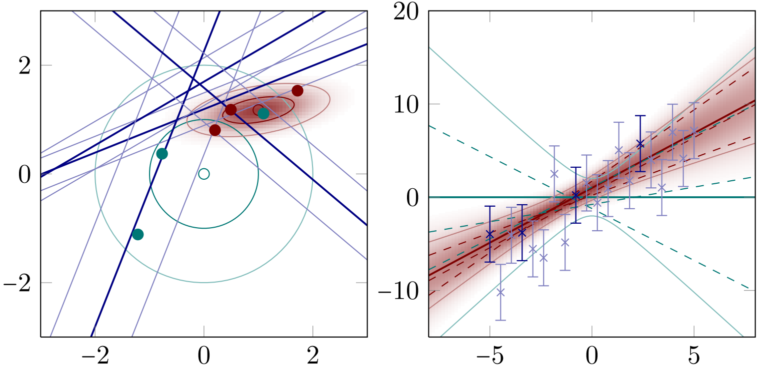
The posterior

$$p(y \mid w, \phi_X) = \mathcal{N}(y; \phi_X^\top w, \sigma^2 I)$$



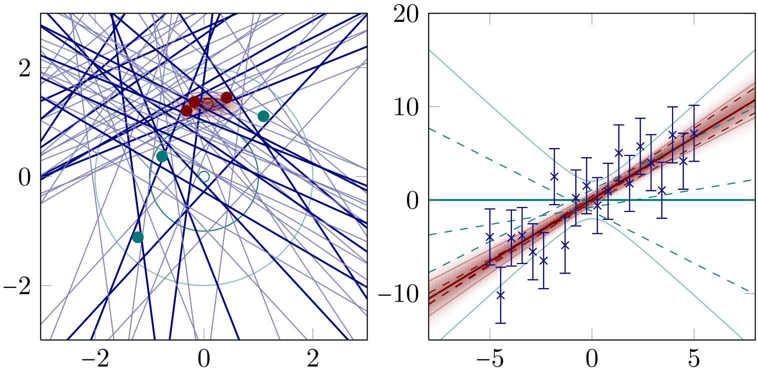
The posterior

$$p(y \mid w, \phi_X) = \mathcal{N}(y; \phi_X^\intercal w, \sigma^2 I)$$



The posterior

$$p(y \mid w, \phi_X) = \mathcal{N}(y; \phi_X^\top w, \sigma^2 I)$$

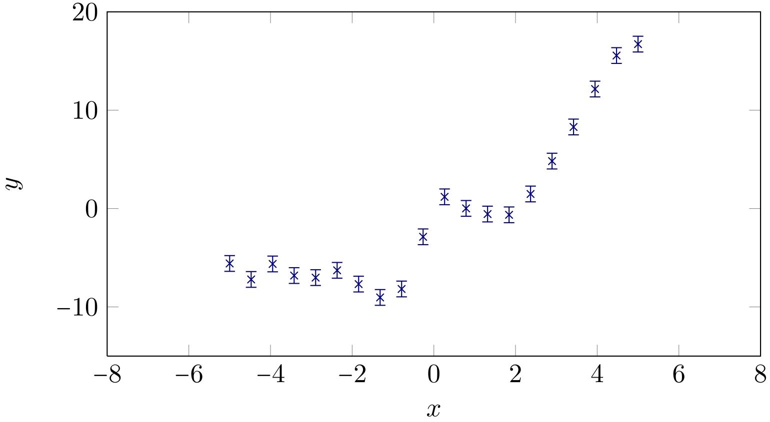


```
% prior on w
F     = 2;                                                          % number of features
phi   = @(a)(bsxfun(@power,a,0:F-1));                               % φ(a) = [1; a]
mu    = zeros(F,1);
Sigma = eye(F);                                                     % p(w) = N(μ, Σ)

% 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) = 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 + ε
phiX  = phi(X);                                                     % features of data
M     = phiX * mu;
kXX   = phiX * Sigma * phiX';                                       % p(f_X) = N(M, k_XX)

G     = kXX + sigma^2 * eye(N);                                     % p(Y) = N(M, k_XX + σ^2 I)
R     = chol(G);                                                    % most expensive step: O(N^3)

kxX   = phix * Sigma * phiX';                                       % 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) = N(m + k_xX (k_XX + σ^2 I)^{-1} (Y - M),
vpost = kxx - A * A';                                   % k_xx - k_xX (k_XX + σ^2 I)^{-1} k_Xx)
spost = bsxfun(@plus,mpost,chol(vpost + 1.0e-8 * eye(n))' * randn(n,3));           % samples

stdpo = sqrt(diag(vpost));                                          % marginal stddev, for plotting
```

A More Realistic Dataset

General Linear Regression

$$f(x) = \phi_x^\top w \qquad ?$$



$$f(x) = w_1 + w_2 x = \phi_x^\intercal w$$

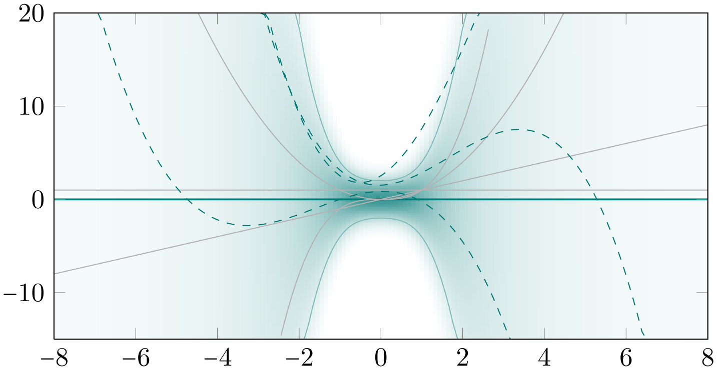
$$\phi_x := \begin{pmatrix} 1 \\ x \end{pmatrix}$$

```
% prior on w
F     = 2;                                                    % number of features
phi   = @(a)(bsxfun(@power,a,0:F-1));                         % φ(a) = [1; a]
mu    = zeros(F,1);
Sigma = eye(F);                                               % p(w) = N(μ, Σ)

% 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) = 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 + ε
phiX  = phi(X);                                               % features of data
M     = phiX * mu;
kXX   = phiX * Sigma * phiX';                                 % p(f_X) = N(M, k_XX)

G     = kXX + sigma^2 * eye(N);                               % p(Y) = N(M, k_XX + σ^2 I)
R     = chol(G);                                              % most expensive step:  O(N^3)

kxX   = phix * Sigma * phiX';                                 % 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) = N(m + k_xX (k_XX + σ^2 I)^-1 (Y − M),
vpost = kxx - A * A';                                         % k_xx − k_xX (k_XX + σ^2 I)^-1 k_Xx)
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

```
phi = @(a)(bsxfun(@power,a,[0:3]));
```

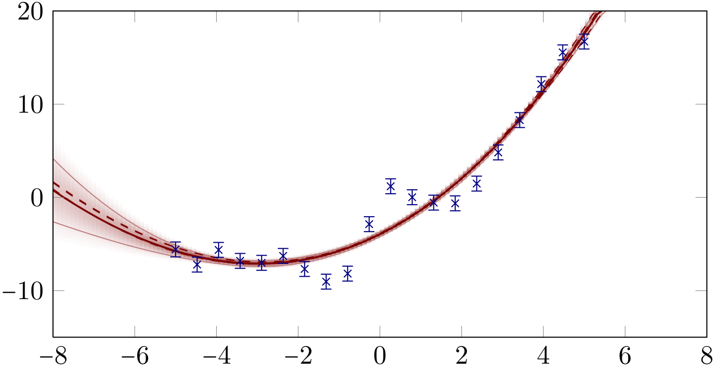
$$f(x) = \phi(x)^\top w \qquad \phi(x) = \begin{pmatrix} 1 & x & x.^2 & x.^3 \end{pmatrix}^\top$$



Cubic Regression

```
phi = @(a)(bsxfun(@power,a,[0:3]));
```

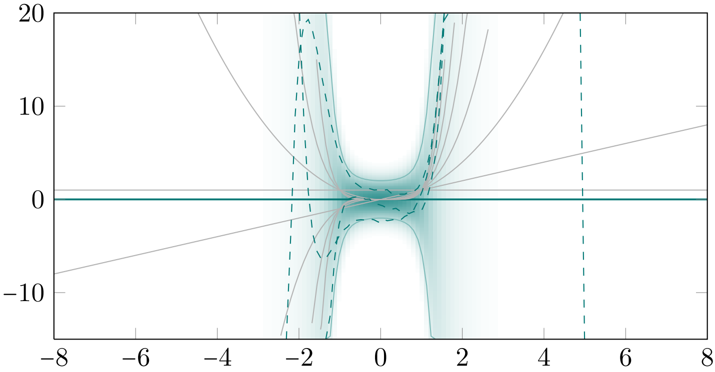
$$f(x) = \phi(x)^\top w \qquad \phi(x) = \begin{pmatrix} 1 & x & x.^2 & x.^3 \end{pmatrix}^\top$$



Septic Regression ?

```
phi = @(a)(bsxfun(@power,a,[0:7]));
```

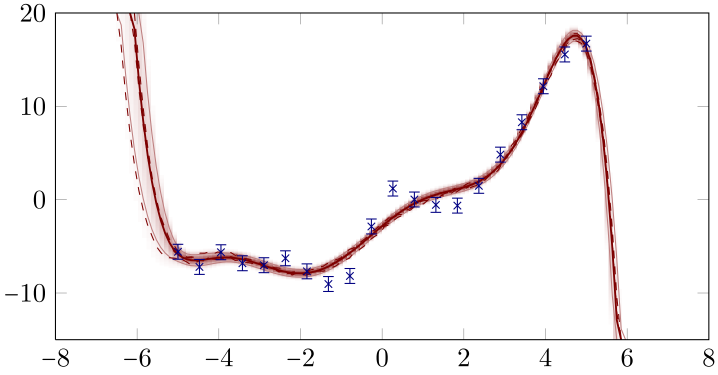
$$f(x) = \phi(x)^\top w \qquad \phi(x) = \begin{pmatrix} 1 & x & x.^2 & \cdots & x.^7 \end{pmatrix}^\top$$



Septic Regression ?

```
phi = @(a)(bsxfun(@power,a,[0:7]));
```

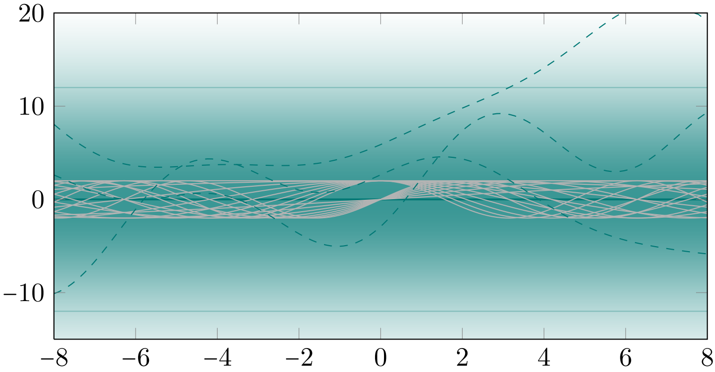
$$f(x) = \phi(x)^\intercal w \qquad \phi(x) = \begin{pmatrix}1 & x & x.^2 & \cdots & x.^7\end{pmatrix}^\intercal$$



Fourier Regression

```
phi = @(a)(2 * [cos(bsxfun(@times,a/8,[0:8])), sin(bsxfun(@times,a/8,[1:8]))]);
```

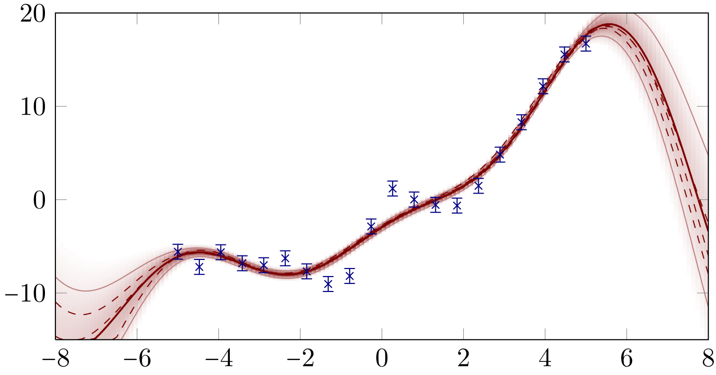
$$\phi(x) = \begin{pmatrix} \cos(x) & \cos(2x) & \cos(3x) & \dots & \sin(x) & \sin(2x) & \dots \end{pmatrix}^\top$$



Fourier Regression

```
phi = @(a)(2 * [cos(bsxfun(@times,a/8,[0:8])), sin(bsxfun(@times,a/8,[1:8]))]);
```

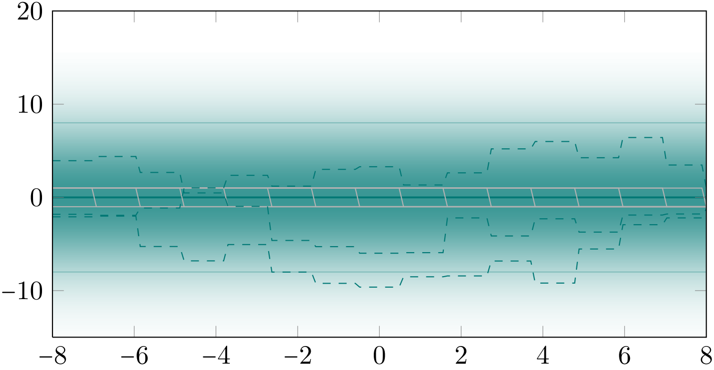
$$\phi(x) = \begin{pmatrix}\cos(x) & \cos(2x) & \cos(3x) & \dots & \sin(x) & \sin(2x) & \dots\end{pmatrix}^\top$$



Step Regression

```
phi = @(a)(-1 + 2 * bsxfun(@lt,a,linspace(-8,8,16)));
```

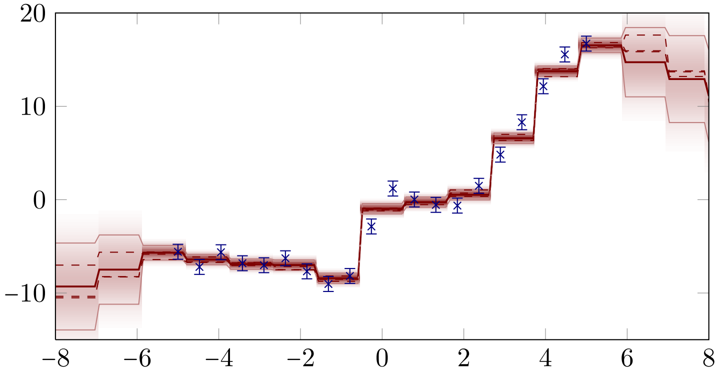
$$\phi(x) = -1 + 2\begin{pmatrix}\theta(x-8) & \theta(8-x) & \theta(x-7) & \theta(7-x) & \dots\end{pmatrix}^{\top}$$



Step Regression

```
phi = @(a)(-1 + 2 * bsxfun(@lt,a,linspace(-8,8,16)));
```

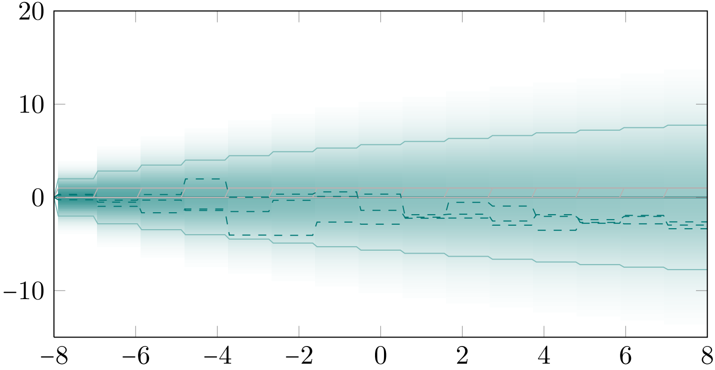
$$\phi(x) = -1 + 2\left(\theta(x-8) \quad \theta(8-x) \quad \theta(x-7) \quad \theta(7-x) \quad \ldots\right)^{\top}$$



Another Kind of Step Regression

```
phi = @(a)(bsxfun(@gt,a,linspace(-8,8,16)));
```

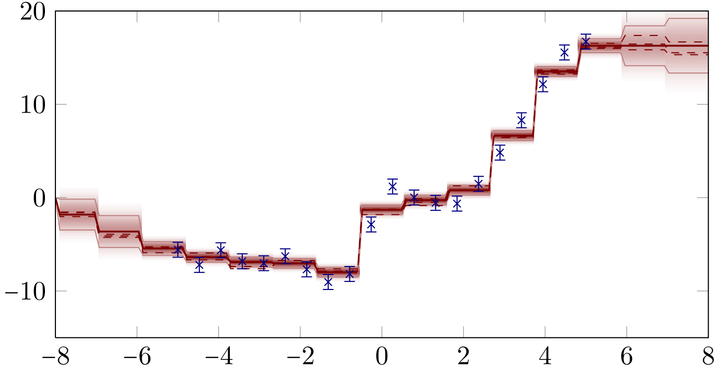
$$\phi(x) = \begin{pmatrix}\theta(x-8) & \theta(8-x) & \theta(x-7) & \theta(7-x) & \dots\end{pmatrix}^\top$$



Another Kind of Step Regression

```
phi = @(a)(bsxfun(@gt,a,linspace(-8,8,16)));
```

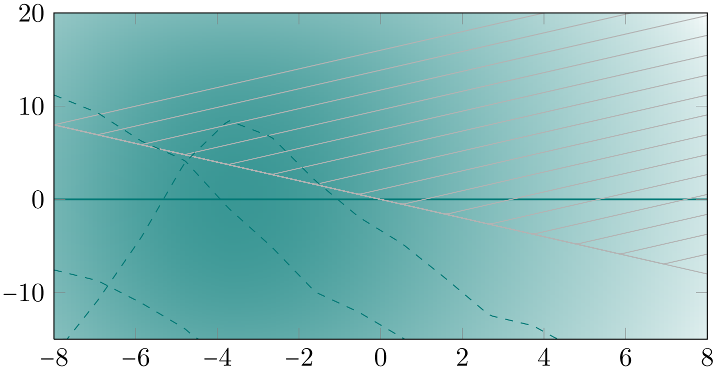
$$\phi(x) = \begin{pmatrix}\theta(x-8) & \theta(8-x) & \theta(x-7) & \theta(7-x) & \dots\end{pmatrix}^\top$$



V Regression

```
phi = @(a)(bsxfun(@minus,abs(bsxfun(@minus,a,linspace(-8,8,16))),linspace(-8,8,16)));
```

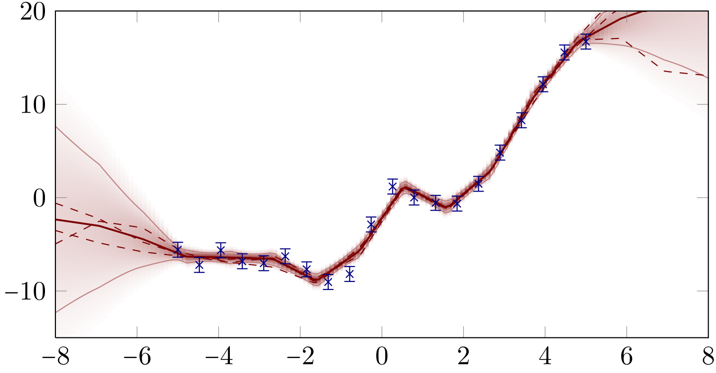
$$\phi(x) = \begin{pmatrix} |x-8|+8 & |x-7|+7 & |x-6|+6 & \dots \end{pmatrix}^\mathsf{T}$$



V Regression

```
phi = @(a)(bsxfun(@minus,abs(bsxfun(@minus,a,linspace(-8,8,16))),linspace(-8,8,16)));
```

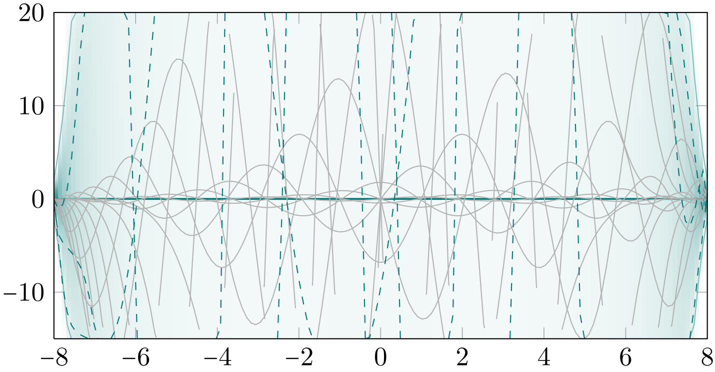
$$\phi(x) = \begin{pmatrix} |x-8|+8 & |x-7|+7 & |x-6|+6 & \dots \end{pmatrix}^{\top}$$



Legendre Regression

```
phi = @(a)(bsxfun(@times,legendre(13,a/8)',0.15.^[0:13]));
```

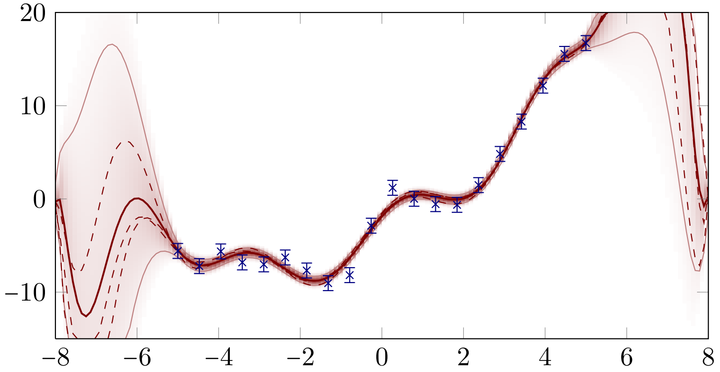
$$\phi(x) = \left(b^0 P_0(x), b^1 P_1(x), \dots, b^{13} P_{13}(x)\right)^\top \qquad P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n}(x^2 - 1)^n$$



Legendre Regression

```
phi = @(a)(bsxfun(@times,legendre(13,a/8)',0.15.^[0:13]));
```

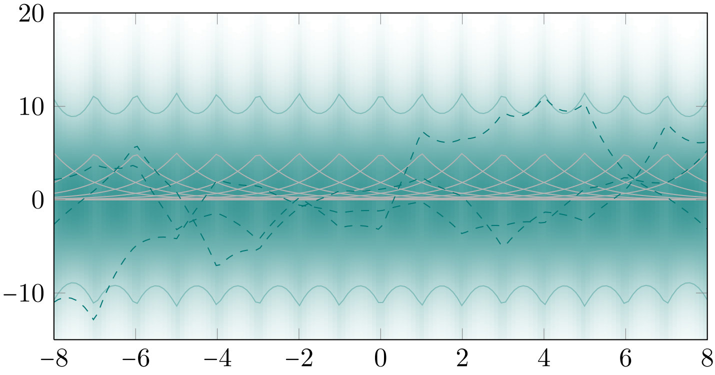
$$\phi(x) = \left(b^0 P_0(x), b^1 P_1(x), \dots, b^{13} P_{13}(x)\right)^\top \qquad P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n}(x^2 - 1)^n$$



Eiffel Tower Regression

```
phi = @(a)(exp(-abs(bsxfun(@minus,a,[-8:1:8]))));
```

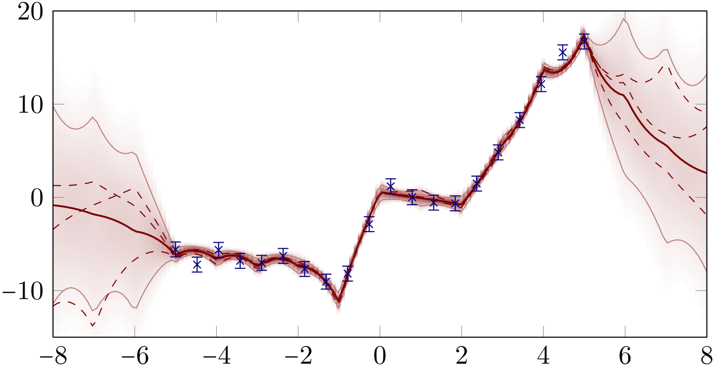
$$\phi(x) = \begin{pmatrix} e^{-|x-8|} & e^{-|x-7|} & e^{-|x-6|} & \dots \end{pmatrix}^{\top}$$



Eiffel Tower Regression

```
phi = @(a)(exp(-abs(bsxfun(@minus,a,[-8:1:8]))));
```

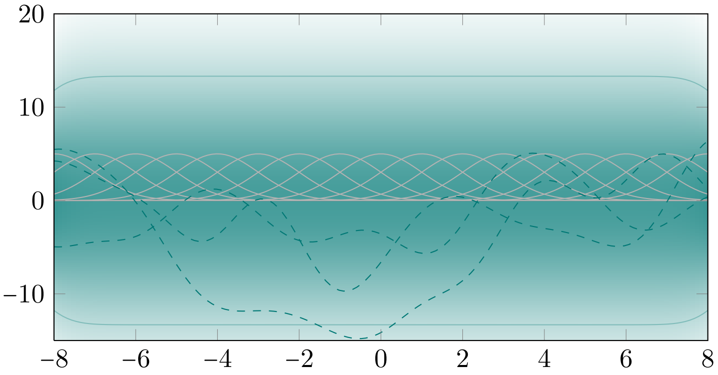
$$\phi(x) = \begin{pmatrix} e^{-|x-8|} & e^{-|x-7|} & e^{-|x-6|} & \dots \end{pmatrix}^\top$$



Bell Curve Regression

```
phi = @(a)(exp(-0.5 * bsxfun(@minus,a,[-8:1:8]).^2));
```

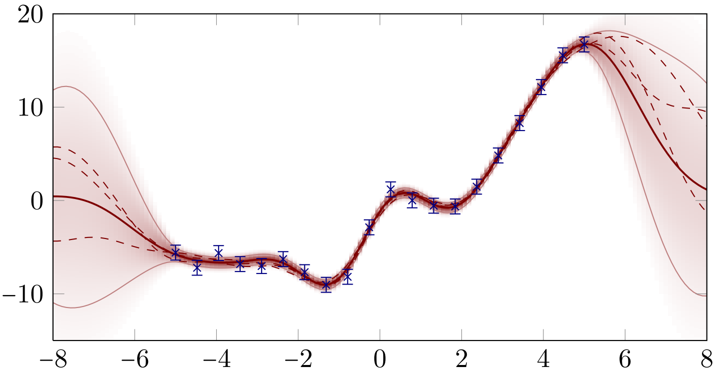
$$\phi(x) = \left(e^{-\frac{1}{2}(x-8)^2} \quad e^{-\frac{1}{2}(x-7)^2} \quad e^{-\frac{1}{2}(x-6)^2} \quad \ldots\right)^\top$$



Bell Curve Regression

```
phi = @(a)(exp(-0.5 * bsxfun(@minus,a,[-8:1:8]).^2));
```

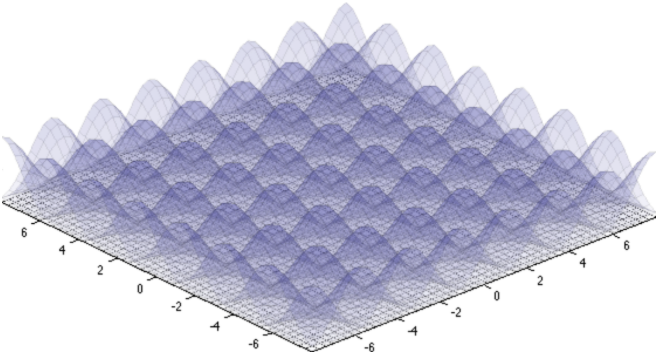
$$\phi(x) = \left(e^{-\frac{1}{2}(x-8)^2} \quad e^{-\frac{1}{2}(x-7)^2} \quad e^{-\frac{1}{2}(x-6)^2} \quad \dots\right)^{\top}$$



Multiple Inputs

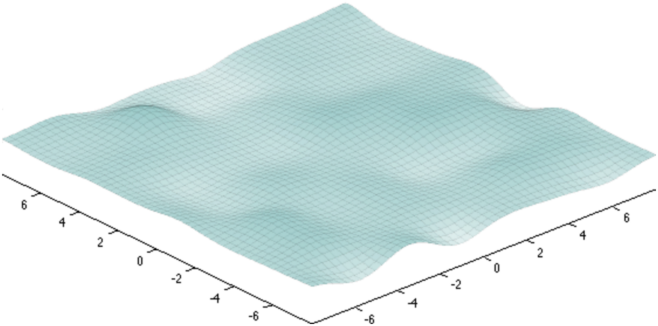
all this works for in multiple dimensions, too

$$\phi : \mathbb{R}^N \to \mathbb{R} \qquad f : \mathbb{R}^N \to \mathbb{R}$$



Multiple Inputs

all this works for in multiple dimensions, too

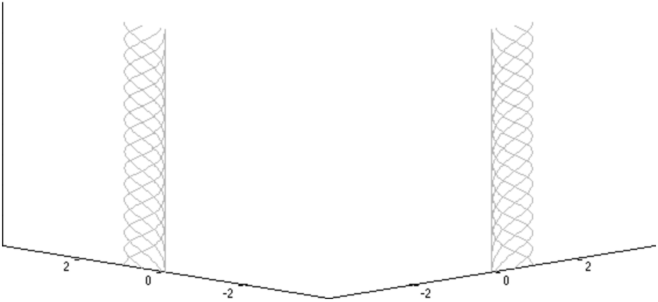


Multiple Outputs

slightly more confusing, but no algebraic problem

$$\phi : \mathbb{R} \to \mathbb{R}^M \qquad f : \mathbb{R} \to \mathbb{R}^M \qquad \operatorname{cov}(f_i(t), f_j(t)) = \sum_\ell \phi_{\ell,i}(t)\phi_{\ell,j}(t')$$

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

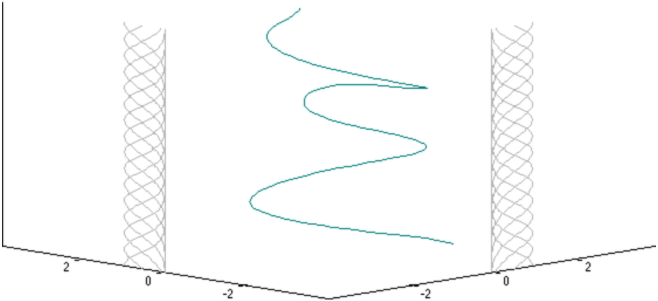


Multiple Outputs

learning paths

$$\phi : \mathbb{R} \to \mathbb{R}^M \qquad f : \mathbb{R} \to \mathbb{R}^M \qquad \operatorname{cov}(f_i(t), f_j(t)) = \sum_{\ell} \phi_{\ell,i}(t)\phi_{\ell,j}(t')$$

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

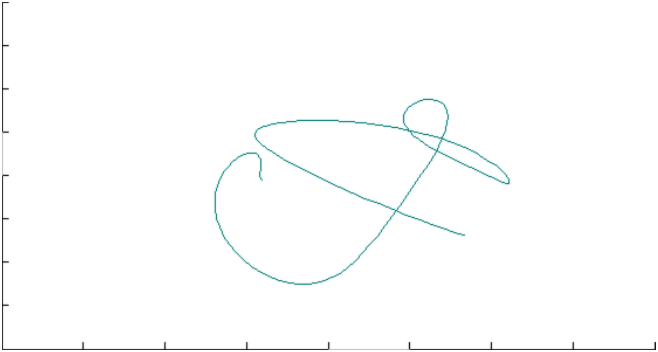


Multiple Outputs

learning paths

$$\phi : \mathbb{R} \to \mathbb{R}^M \qquad f : \mathbb{R} \to \mathbb{R}^M \qquad \operatorname{cov}(f_i(t), f_j(t)) = \sum_\ell \phi_{\ell,i}(t)\phi_{\ell,j}(t')$$

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



- Gaussians link *inference* and *linear algebra*
- linear weights with features model *functions*
- in fact, the number of features can be infinite!

How many features should we use?

let's look at that algebra again

$$p(f_x \mid y, \phi_X) = \mathcal{N}(f_x; \phi_x^\top \mu + \phi_x^\top \Sigma \phi_X (\phi_X^\top \Sigma \phi_X + \sigma^2 I)^{-1} (y - \phi_X^\top \mu),$$
$$\phi_x^\top \Sigma \phi_x - \phi_x^\top \Sigma \phi_X (\phi_X^\top \Sigma \phi_X + \sigma^2 I)^{-1} \phi_X^\top \Sigma \phi_x)$$

- there's no lonely ϕ in there
- all objects involving ϕ are of the form
  - $\phi^\top \mu$ — the mean function
  - $\phi^\top \Sigma \phi$ — the kernel
- once these are known, cost is independent of the number of features
- remember the code:

```
M     = phiX * mu;
m     = phix * mu;
kXX   = phiX * Sigma * phiX';          % p(f_X) = N(M, k_XX)
kxx   = phix * Sigma * phix';          % p(f_x) = N(m, k_xx)

kxX   = phix * Sigma * phiX';          % cov(f_x, f_X) = k_xX
```

```
% prior on w
F     = 2;                                                  % number of features
phi   = @(a)(bsxfun(@power,a,0:F-1));                       % φ(a) = [1; a]
mu    = zeros(F,1);
Sigma = eye(F);                                             % p(w) = N(μ, Σ)

% 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) = 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 + ε
phiX  = phi(X);                                             % features of data
M     = phiX * mu;
kXX   = phiX * Sigma * phiX';                               % p(f_X) = N(M, k_XX)

G     = kXX + sigma^2 * eye(N);                             % p(Y) = N(M, k_XX + σ^2 I)
R     = chol(G);                                            % most expensive step:  O(N^3)

kxX   = phix * Sigma * phiX';                               % 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) = N(m + k_xX (k_XX + σ^2 I)^{-1} (Y - M),
vpost = kxx - A * A';                                       % k_xx - k_xX (k_XX + σ^2 I)^{-1} k_Xx)
spost = bsxfun(@plus,mpost,chol(vpost + 1.0e-8 * eye(n))' * randn(n,3));      % samples

stdpo = sqrt(diag(vpost));                                  % 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 = linspace(-6,6,n)';                           % 'test' points
m     = mu(x);
kxx   = k(x,x);                                               % p(f_x) = 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 + ε
M     = mu(X);
kXX   = k(X,X);                                               % p(f_X) = N(M, k_XX)

G     = kXX + sigma^2 * eye(N);                               % p(Y) = N(M, k_XX + σ^2 I)
R     = chol(G);                                              % most expensive step: O(N^3)

kxX   = k(x,X);                                               % 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) = N(m + k_xX (k_XX + σ^2 I)^{-1} (Y - M),
vpost = kxx - A * A';                                   % k_xx - k_xX (k_XX + σ^2 I)^{-1} k_Xx)
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

an example

DJC MacKay, 1998

- For simplicity, let's fix $\Sigma = \frac{\sigma^2(c_{\max}-c_{\min})}{F} I$
- The elements of $\phi_x^\top \Sigma \phi_x$ are

$$\phi(x_i)^\top \Sigma \phi(x_j) = \frac{\sigma^2(c_{\max} - c_{\min})}{F} \sum_{\ell=1}^{F} \phi_\ell(x_i)\phi_\ell(x_j)$$

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

$$\phi_\ell(x) = \exp\left(-\frac{(x - c_\ell)^2}{2\lambda^2}\right)$$

$$\begin{aligned}
&\phi(x_i)^\top \Sigma \phi(x_j) \\
&= \frac{\sigma^2(c_{\max} - c_{\min})}{F} \sum_{\ell=1}^{F} \exp\left(-\frac{(x_i - c_\ell)^2}{2\lambda^2}\right) \exp\left(-\frac{(x_j - c_\ell)^2}{2\lambda^2}\right) \\
&= \frac{\sigma^2(c_{\max} - c_{\min})}{F} \exp\left(-\frac{(x_i - x_j)^2}{4\lambda^2}\right) \sum_{\ell}^{F} \exp\left(-\frac{(c_\ell - \frac{1}{2}(x_i + x_j))^2}{\lambda^2}\right)
\end{aligned}$$

Features are cheap, so let's use a lot

an example

DJC MacKay, 1998

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

- now increase F, such that # of features in δc becomes $\frac{F \cdot \delta c}{(c_{\max} - c_{\min})}$

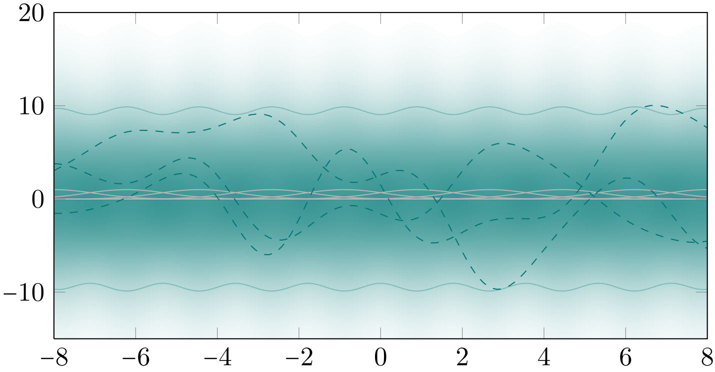
$$\phi(x_i)^\top \Sigma \phi(x_j) \to$$
$$\sigma^2 \exp\left(-\frac{(x_i - x_j)^2}{4\lambda^2}\right) \int_{c_{\min}}^{c_{\max}} \exp\left(-\frac{(c - \frac{1}{2}(x_i + x_j))^2}{\lambda^2}\right) \mathrm{d}c$$

- let $c_{\min} \to -\infty$, $c_{\max} \to \infty$

$$\phi(x_i)^\top \Sigma \phi(x_j) \to \sqrt{2\pi}\lambda\sigma^2 \exp\left(-\frac{(x_i - x_j)^2}{4\lambda^2}\right)$$

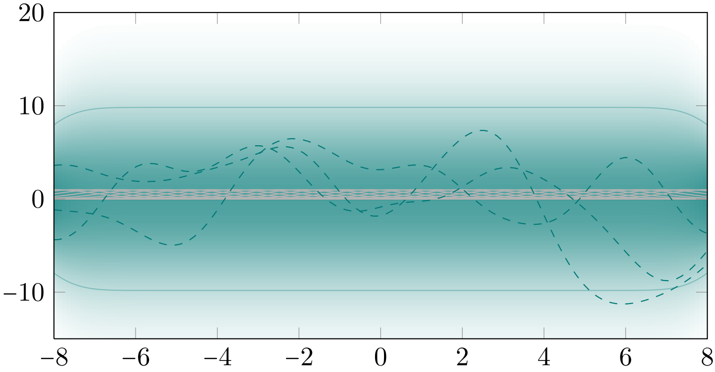
Exponentiated Squares

```
phi = @(a)(exp(-0.5 * bsxfun(@minus,a,linspace(-8,8,10)).^2 ./ell.^2));
```



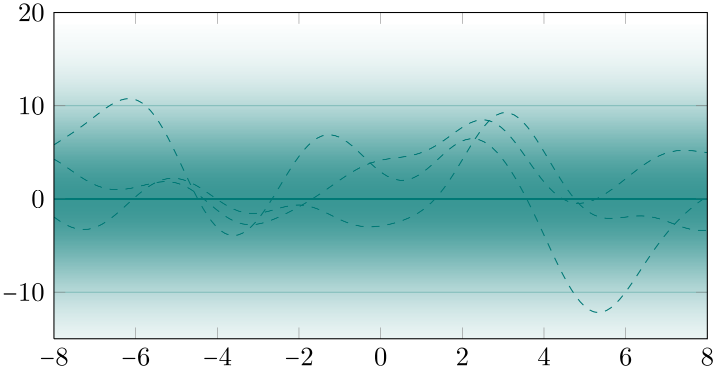
Exponentiated Squares

```
phi = @(a)(exp(-0.5 * bsxfun(@minus,a,linspace(-8,8,30)).^2 ./ell.^2));
```



Exponentiated Squares

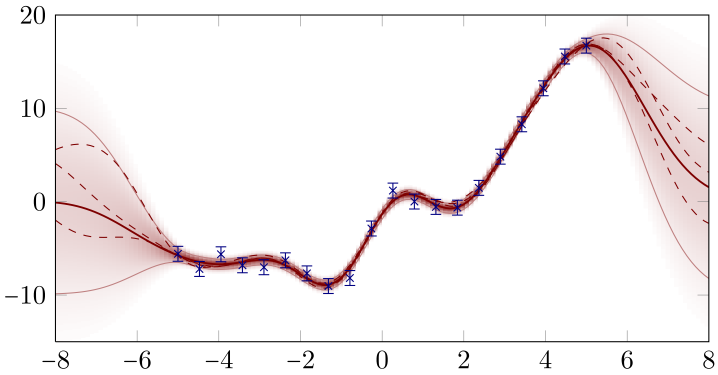
```
k = @(a,b)(5*exp(-0.25*bsxfun(@minus,a,b').^2));
```



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

Exponentiated Squares

```
k = @(a,b)(5*exp(-0.25*bsxfun(@minus,a,b').^2));
```



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

What just happened?

kernelization to infinitely many features

Definition

A function $k : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$ is a Mercer kernel if, for any finite collection $X = [x_1, \dots, x_N]$, 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

$$k(x, x') = \sum\!\!\!\!\!\!\int \phi_\ell(x)\phi_\ell(x')\, d\ell$$

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

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

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

What just happened?

Gaussian process priors

Definition

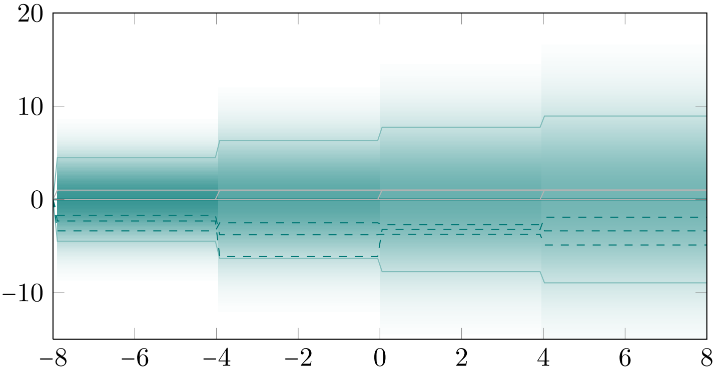
A function $k : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$ is a Mercer kernel if, for any finite collection $X = [x_1, \dots, x_N]$, the matrix $k_{XX} \in \mathbb{R}^{N \times N}$ with elements $k_{XX,(i,j)} = k(x_i, x_j)$ is positive semidefinite.

Definition

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

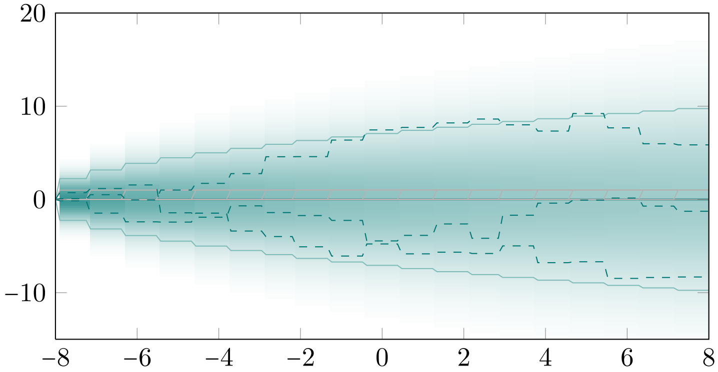
Those step functions

```
phi = @(a)(bsxfun(@gt,a,linspace(-8,8,5))./sqrt(5));
```



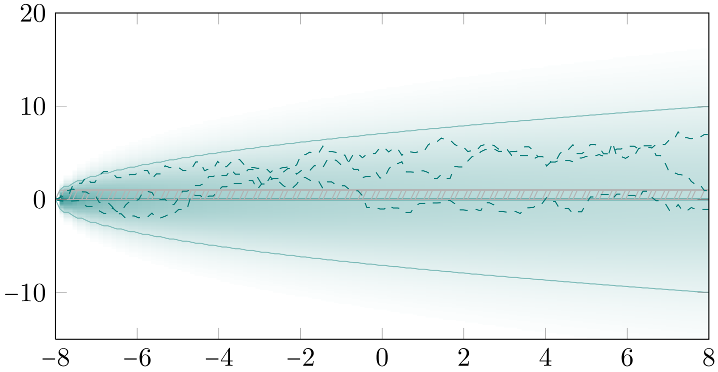
Those step functions

```
phi = @(a)(bsxfun(@gt,a,linspace(-8,8,20))./sqrt(20));
```



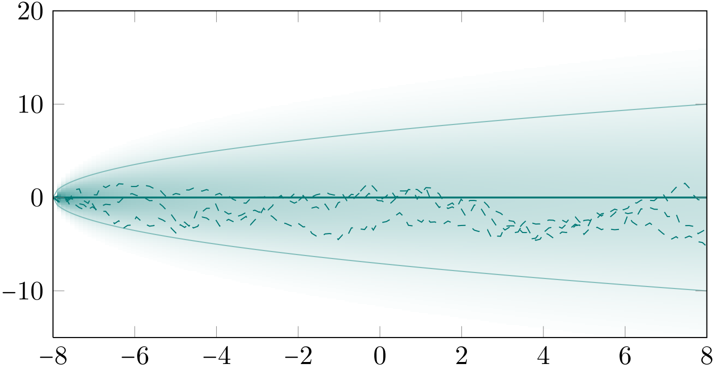
Those step functions

```
phi = @(a)(bsxfun(@gt,a,linspace(-8,8,100))./sqrt(100));
```



Those step functions

```
k = @(a,b)(theta.^2 * bsxfun(@min,a+8,b'+8)/16);
```

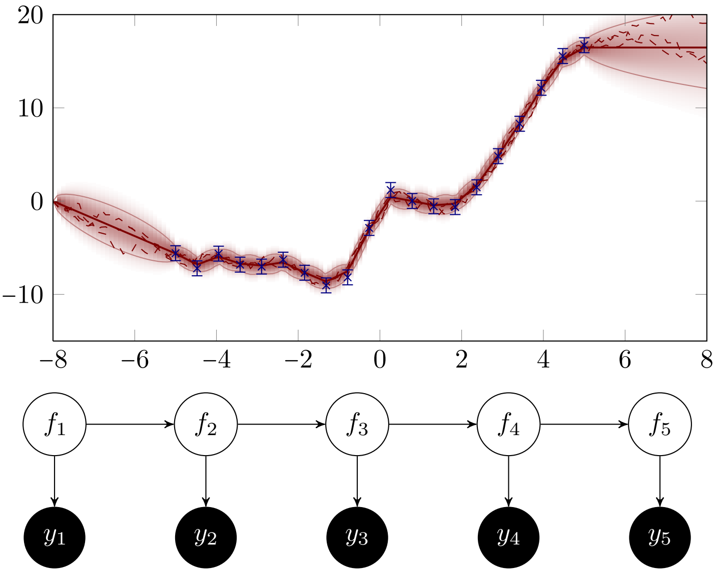


$$\mathrm{cov}(f_{x_i}, f_{x_j}) = \int_{c_{\min}}^{\infty} \theta(x_i - c)\theta(x_j - c)\, \mathrm{d}c = \min(x_i, x_j) - c_{\min}$$

- aka. the Wiener process

Those step functions

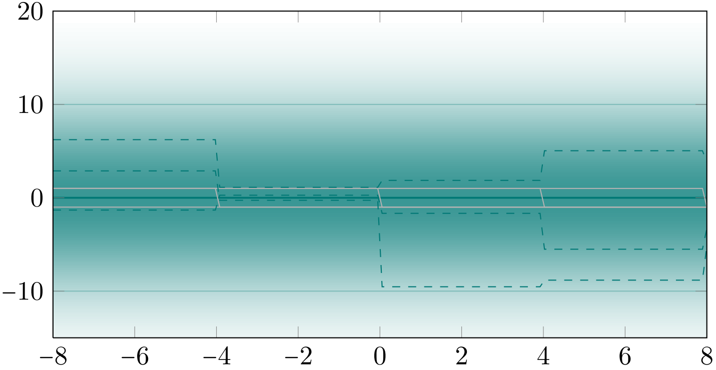
```
k = @(a,b)(theta.^2 * bsxfun(@min,a+8,b'+8)/16);
```



Those other step-functions

```
phi = @(a)(-1 + 2 * bsxfun(@lt,a,linspace(-8,8,5)));
```

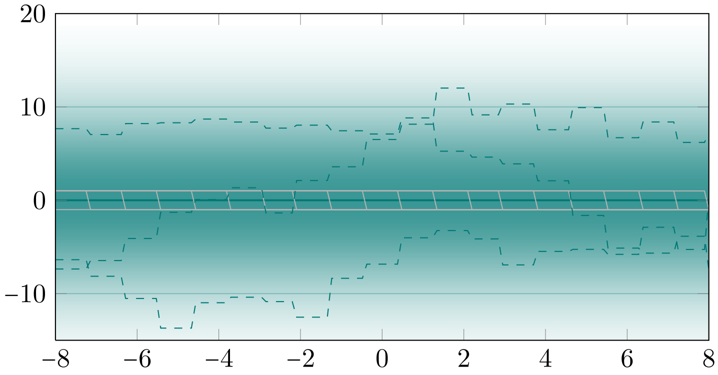
Wahba, 1990



Those other step-functions

```
phi = @(a)(-1 + 2 * bsxfun(@lt,a,linspace(-8,8,20)));
```

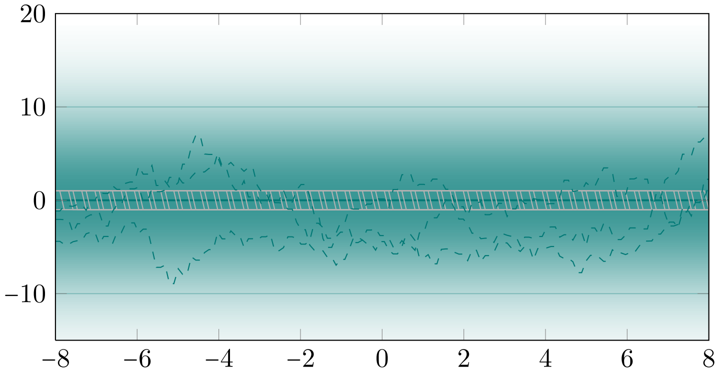
Wahba, 1990



Those other step-functions

```
phi = @(a)(-1 + 2 * bsxfun(@lt,a,linspace(-8,8,100)));
```

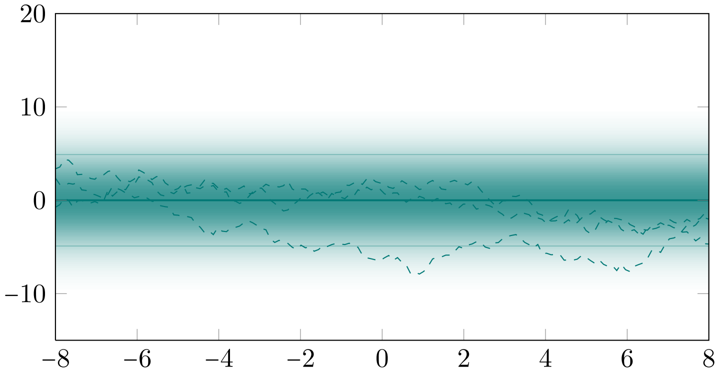
Wahba, 1990



Those other step-functions

```
k = @(a,b)((1 + c - 2 * c * abs(bsxfun(@minus,a,b')/16)));
```

Wahba, 1990



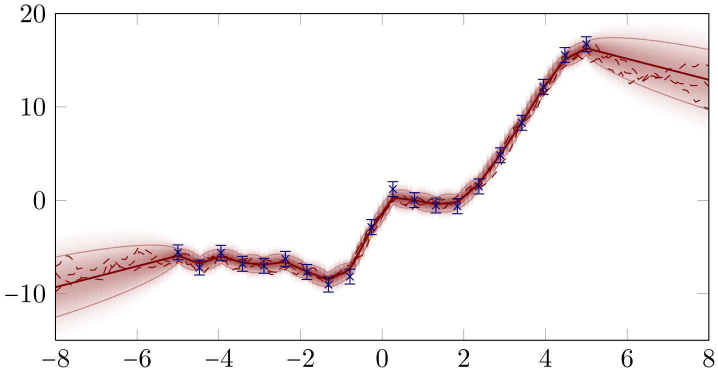
$$\operatorname{cov}(f_{x_i}, f_{x_j}) = 1 + b\int_0^1 (2\theta(x_i - c) - 1)(2\theta(x_j - c) - 1)\,\mathrm{d}c = 1 + b - 2b|x_i - x_j|$$

- aka. linear splines

Those other step-functions

```
k = @(a,b)((1 + c - 2 * c * abs(bsxfun(@minus,a,b')/16)));
```

Wahba, 1990



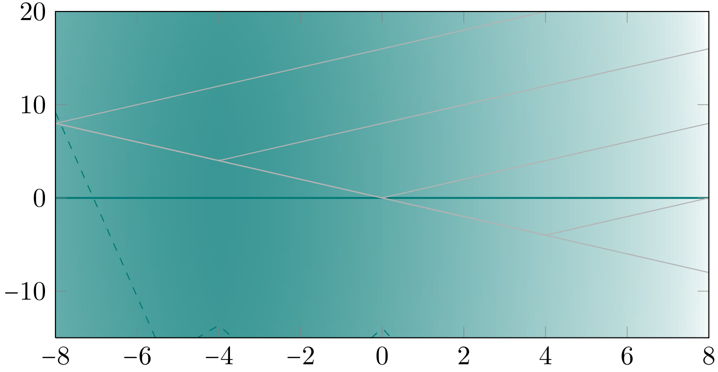
$$\operatorname{cov}(f_{x_i}, f_{x_j}) = 1 + b\int_0^1 (2\theta(x_i - c) - 1)(2\theta(x_j - c) - 1)\,\mathrm{d}c = 1 + b - 2b|x_i - x_j|$$

- aka. linear splines

Those linear features

Wahba, 1990

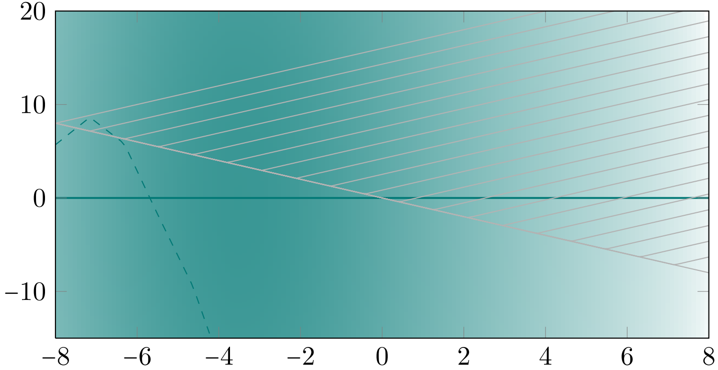
```
phi = @(a)(bsxfun(@minus,abs(bsxfun(@minus,a,linspace(-8,8,5))),linspace(-8,8,5)));
```



Those linear features

Wahba, 1990

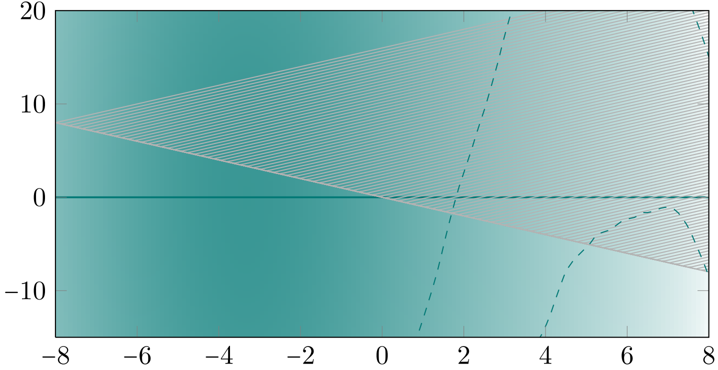
```
phi = @(a)(bsxfun(@minus,abs(bsxfun(@minus,a,linspace(-8,8,20))),linspace(-8,8,20)));
```



Those linear features

Wahba, 1990

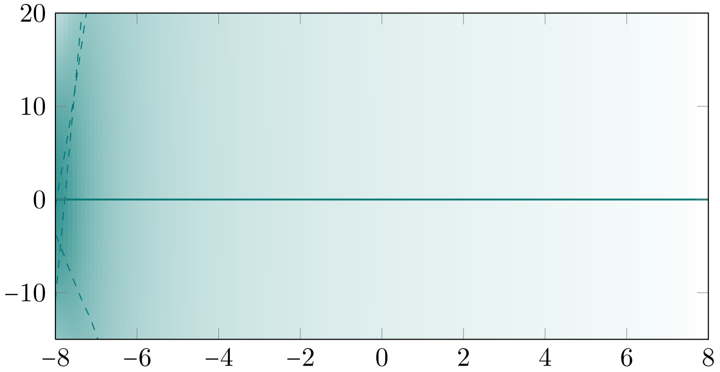
```
phi =
@(a)(bsxfun(@minus,abs(bsxfun(@minus,a,linspace(-8,8,100))),linspace(-8,8,100)));
```



Those linear features

Wahba, 1990

```
k = @(a,b)(theta.^2 * (1 + (1+c) * bsxfun(@times,a+8,b'+8)./16 + c ./ 3 *
(abs(bsxfun(@minus,a,b')/16).^3 - bsxfun(@plus,((a+8)./16).^3,((b'+8)./16).^3))));
```



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

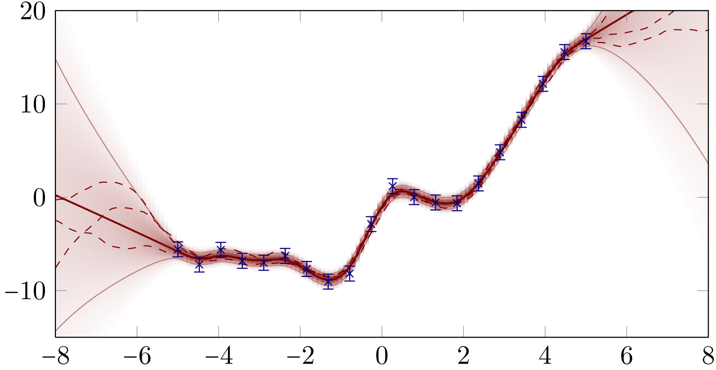
$$= 1 + (1+b)x_i x_j + \frac{b}{3}(|x_i - x_j|^3 - x^3 - y^3)$$

aka. cubic splines

Those linear features

Wahba, 1990

```
k = @(a,b)(theta.^2 * (1 + (1+c) * bsxfun(@times,a+8,b'+8)./16 + c ./ 3 *
(abs(bsxfun(@minus,a,b')/16).^3 - bsxfun(@plus,((a+8)./16).^3,((b'+8)./16).^3))));
```



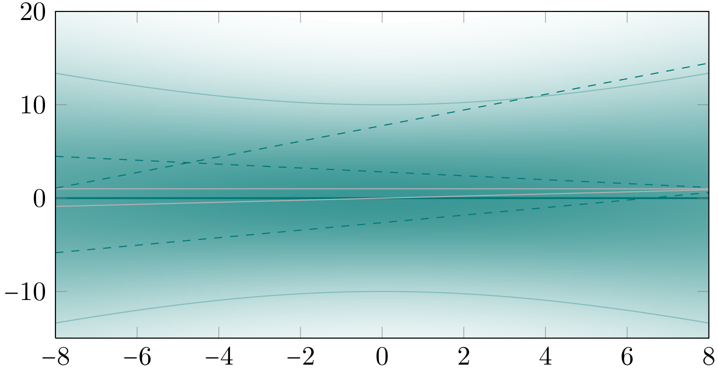
$$
\begin{aligned}
\operatorname{cov}(f_{x_i}, f_{x_j}) &= 1 + x_i x_j + b \int_0^1 (|x_i - c| - c)(|x_j - c| - c)\,\mathrm{d}c \\
&= 1 + (1+b)x_i x_j + \frac{b}{3}(|x_i - x_j|^3 - x^3 - y^3)
\end{aligned}
$$

aka. cubic splines

Exponentially suppressed polynomials

```
phi = @(a)(bsxfun(@times,bsxfun(@power,a./9,[0:1]),c.^[0:1]));
```

Minka, 2000

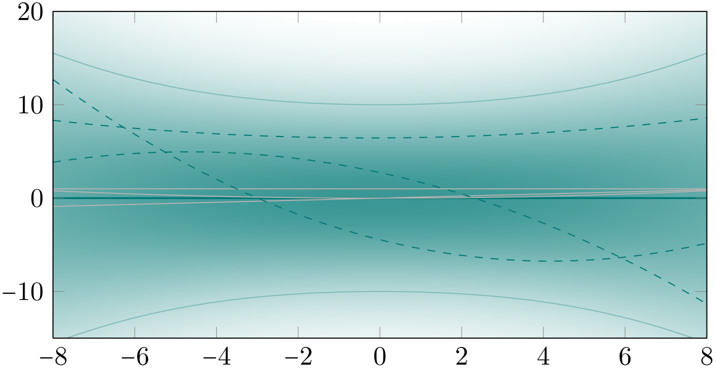


$$\operatorname{cov}(f_{x_i}, f_{x_j}) = \sum_{\ell=0}^{1} b^\ell x_i^\ell x_j^\ell \qquad 0 \leq b \leq 1 \qquad -1 < x_i, x_j < 1$$

Exponentially suppressed polynomials

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

Minka, 2000

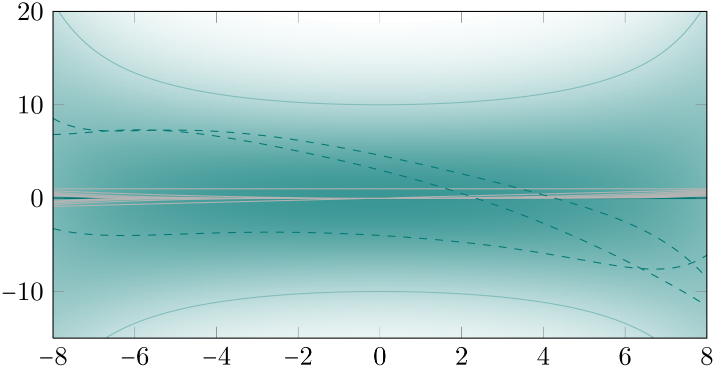


$$\operatorname{cov}(f_{x_i}, f_{x_j}) = \sum_{\ell=0}^{2} b^\ell x_i^\ell x_j^\ell \qquad 0 \le b \le 1 \qquad -1 < x_i, x_j < 1$$

Exponentially suppressed polynomials

```
phi = @(a)(bsxfun(@times,bsxfun(@power,a./9,[0:10]),c.^[0:10]));
```

Minka, 2000

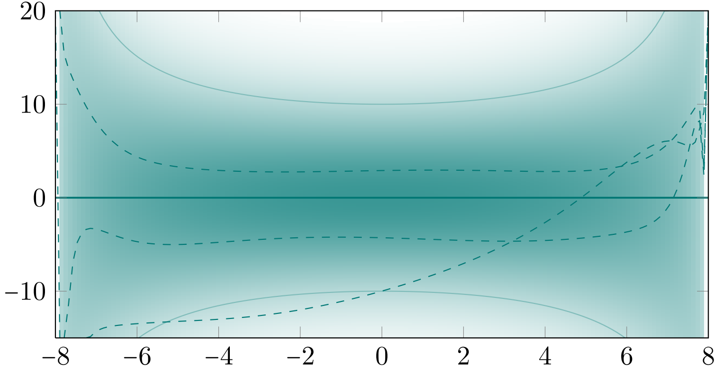


$$\operatorname{cov}(f_{x_i}, f_{x_j}) = \sum_{\ell=0}^{10} b^\ell x_i^\ell x_j^\ell \qquad 0 \le b \le 1 \qquad -1 < x_i, x_j < 1$$

Exponentially suppressed polynomials

```
k = @(a,b)(theta.^2 .* 1./(1-c*bsxfun(@times,a./8,b'./8)));
```

Minka, 2000

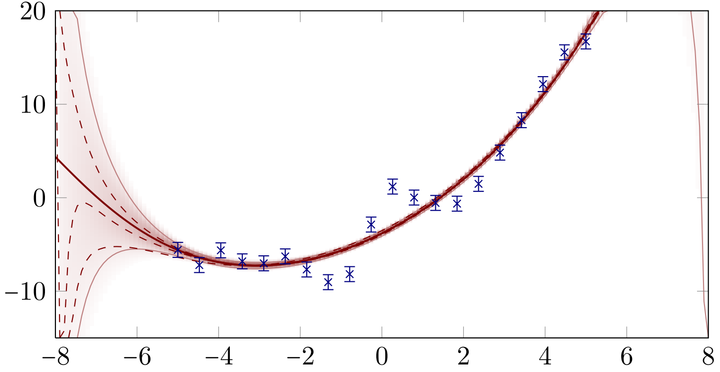


$$\mathrm{cov}(f_{x_i}, f_{x_j}) = \sum_{\ell=0}^{\infty} b^\ell x_i^\ell x_j^\ell = \frac{1}{1 - b x_i x_j} \qquad 0 \le b \le 1 \qquad -1 < x_i, x_j < 1$$

Exponentially suppressed polynomials

```
k = @(a,b)(theta.^2 .* 1./(1-c*bsxfun(@times,a./8,b'./8)));
```

Minka, 2000

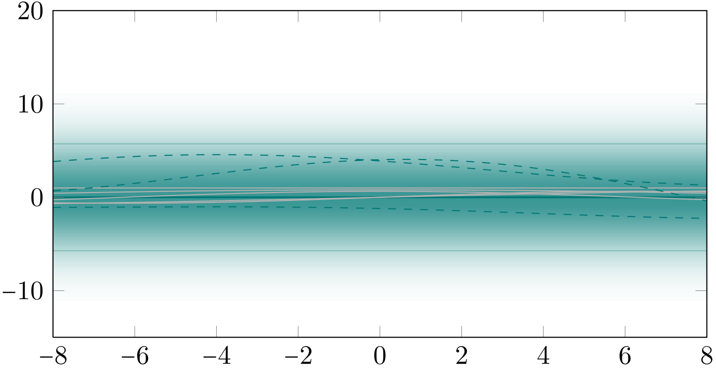


$$\operatorname{cov}(f_{x_i}, f_{x_j}) = \sum_{\ell=0}^{\infty} b^\ell x_i^\ell x_j^\ell = \frac{1}{1 - b x_i x_j} \qquad 0 \leq b \leq 1 \qquad -1 < x_i, x_j < 1$$

Exponentially decaying periodic features

T. Minka, 2000

```
phi = @(a)([bsxfun(@times,cos(bsxfun(@times,a/8,[0:2])),c.^[0:2]), ...
bsxfun(@times,sin(bsxfun(@times,a/8,[1:2])),c.^[1:2])]);
```



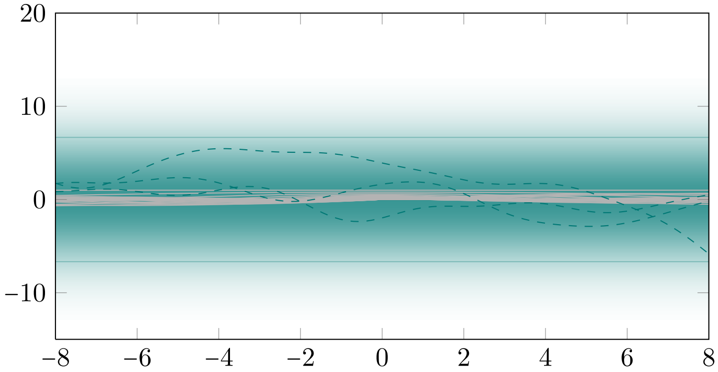
$$\operatorname{cov}(f_{x_i}, f_{x_j}) = 1 + \sum_{\ell=0}^{2} b^{\ell} (\cos(2\pi\ell x_i) \cos(2\pi\ell x_j) + \sin(2\pi\ell x_i) \sin(2\pi\ell x_j))$$

$$0 \leq b \leq 1$$

Exponentially decaying periodic features

T. Minka, 2000

```
phi = @(a)([bsxfun(@times,cos(bsxfun(@times,a/8,[0:20])),c.^[0:20]), ...
bsxfun(@times,sin(bsxfun(@times,a/8,[1:20])),c.^[1:20])]);
```



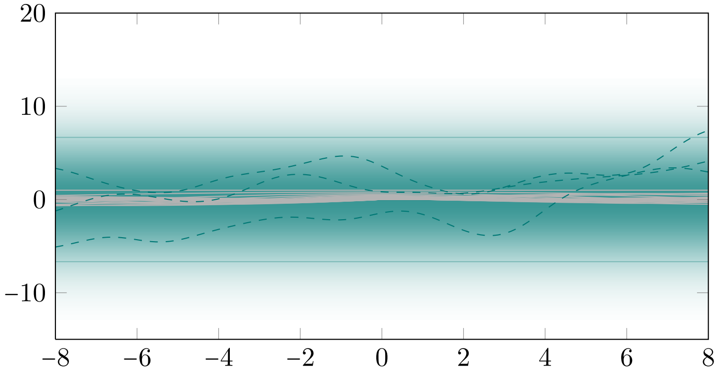
$$\mathrm{cov}(f_{x_i}, f_{x_j}) = 1 + \sum_{\ell=0}^{20} b^{\ell}\left(\cos(2\pi\ell x_i)\cos(2\pi\ell x_j) + \sin(2\pi\ell x_i)\sin(2\pi\ell x_j)\right)$$

$$0 \leq b \leq 1$$

Exponentially decaying periodic features

T. Minka, 2000

```
phi = @(a)([bsxfun(@times,cos(bsxfun(@times,a/8,[0:50])),c.^[0:50]), ...
bsxfun(@times,sin(bsxfun(@times,a/8,[1:50])),c.^[1:50])]);
```



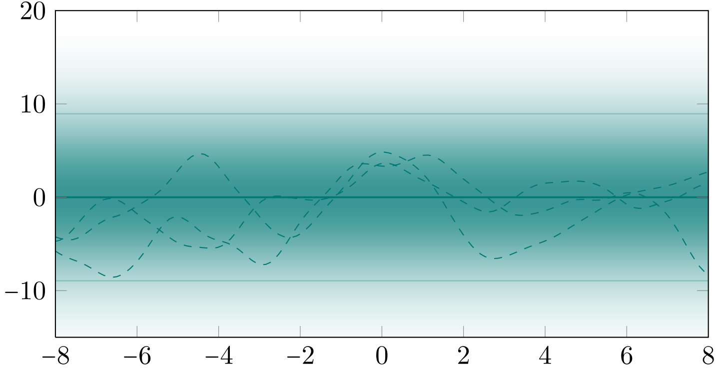
$$\operatorname{cov}(f_{x_i}, f_{x_j}) = 1 + \sum_{\ell=0}^{50} b^\ell \left(\cos(2\pi\ell x_i)\cos(2\pi\ell x_j) + \sin(2\pi\ell x_i)\sin(2\pi\ell x_j)\right)$$

$$0 \leq b \leq 1$$

Exponentially decaying periodic features

T. Minka, 2000

```
k = @(a,b)(theta.^2 .* 0.5 .* (1 + (1 - c.^2) ./ (1 + c.^2 ...
- 2 * c * cos(bsxfun(@minus,a,b')/8))));
```



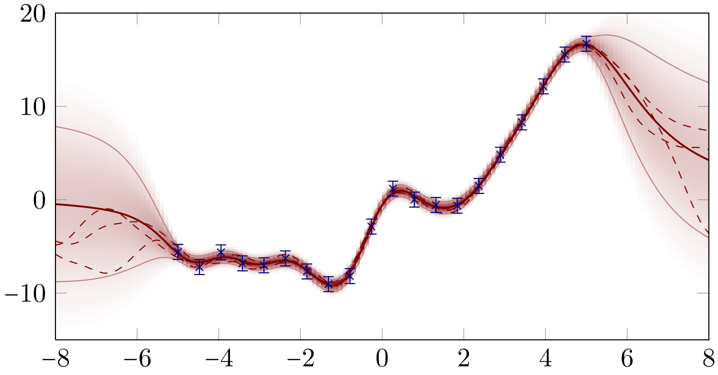
$$\mathrm{cov}(f_{x_i}, f_{x_j}) = 1 + \sum_{\ell=0}^{\infty} b^{\ell}(\cos(2\pi\ell x_i)\cos(2\pi\ell x_j) + \sin(2\pi\ell x_i)\sin(2\pi\ell x_j))$$

$$= \frac{1}{2} + \frac{(1-b^2)/2}{1 + b^2 - 2b\cos(2\pi(x_i - x_j))} \qquad 0 \le b \le 1$$

Exponentially decaying periodic features

T. Minka, 2000

```
k = @(a,b)(theta.^2 .* 0.5 .* (1 + (1 - c.^2) ./ (1 + c.^2 ...
- 2 * c * cos(bsxfun(@minus,a,b')/8))));
```

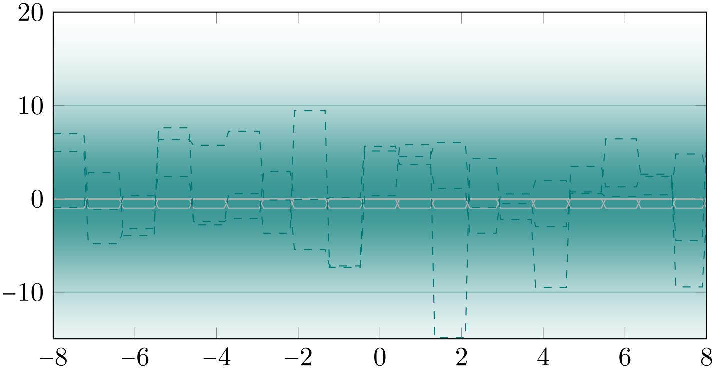


$$\operatorname{cov}(f_{x_i}, f_{x_j}) = 1 + \sum_{\ell=0}^{\infty} b^\ell \left(\cos(2\pi\ell x_i)\cos(2\pi\ell x_j) + \sin(2\pi\ell x_i)\sin(2\pi\ell x_j)\right)$$

$$= \frac{1}{2} + \frac{(1-b^2)/2}{1+b^2-2b\cos(2\pi(x_i-x_j))} \qquad 0 \leq b \leq 1$$

"White Noise"

the "limit" of block functions

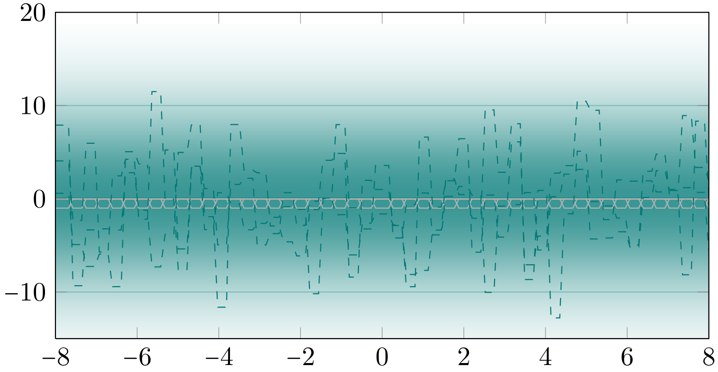


$$\lim_{\epsilon \to 0} \int \mathbb{I}(|x_i - c| < \epsilon)\mathbb{I}(|x_j - c| < \epsilon)\, \mathrm{d}c = \delta(x_i - x_j)$$

- 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

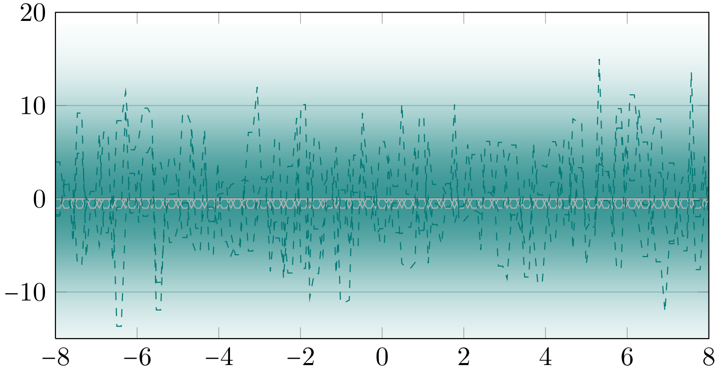


$$\lim_{\epsilon \to 0} \int \mathbb{I}(|x_i - c| < \epsilon)\mathbb{I}(|x_j - c| < \epsilon)\,\mathrm{d}c = \delta(x_i - x_j)$$

- 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

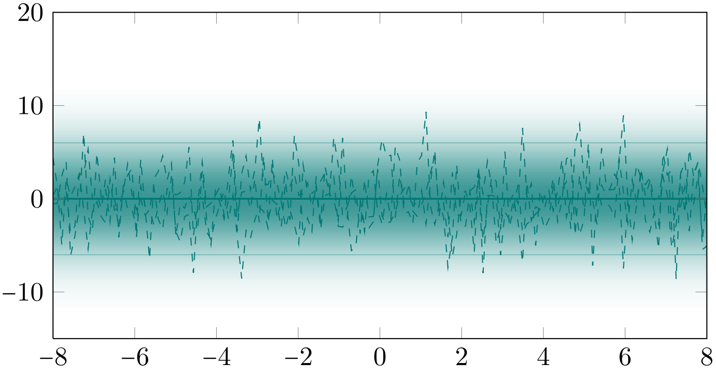


$$\lim_{\epsilon \to 0} \int \mathbb{I}(|x_i - c| < \epsilon)\mathbb{I}(|x_j - c| < \epsilon)\, \mathrm{d}c = \delta(x_i - x_j)$$

- 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

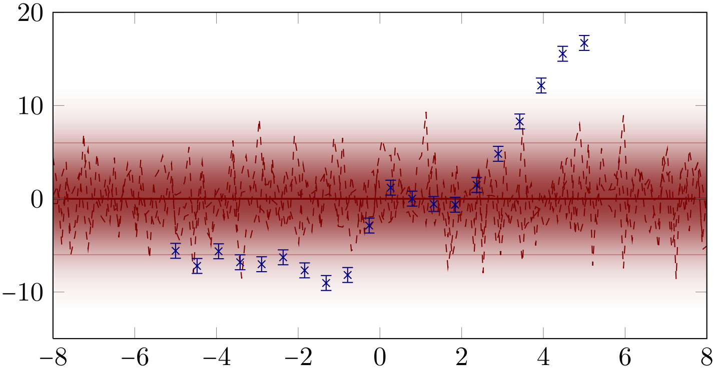


$$\lim_{\epsilon \to 0} \int \mathbb{I}(|x_i - c| < \epsilon)\mathbb{I}(|x_j - c| < \epsilon) \, \mathrm{d}c = \delta(x_i - x_j)$$

- 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



$$\lim_{\epsilon \to 0} \int \mathbb{I}(|x_i - c| < \epsilon)\mathbb{I}(|x_j - c| < \epsilon)\, \mathrm{d}c = \delta(x_i - x_j)$$

- 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

- Gaussians link *inference* and *linear algebra*
- linear weights with features model *functions*
- in fact, the number of features can be infinite!
- kernels can be
  - *output scaled*
  - *input scaled*
  - *added*
  - *multiplied*

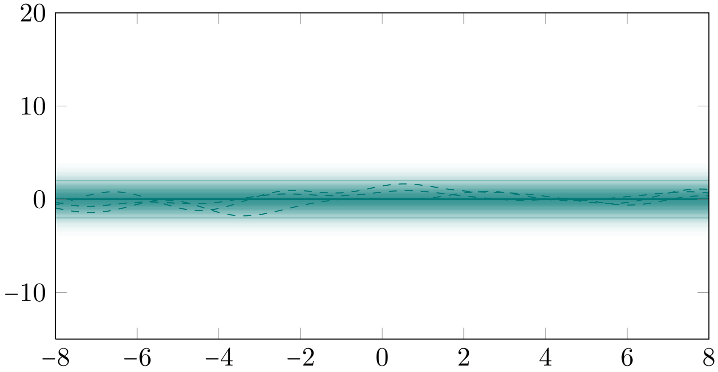
  to get more expressive models

Scaling Outputs

```
k = @(a,b)(1.^2 * exp(-(bsxfun(@minus,a./2,b'./2)).^2));
```

$$v^\top k v \geq 0 \; \forall v \quad \Rightarrow \quad v^\top \theta^2 k v = \theta^2 v^\top k v \geq 0 \; \forall v$$

$$p(f) = \mathcal{GP}(f; \mu, k) \quad \Rightarrow \quad \mathrm{var}[f(x)] = \theta^2 k(x, x)$$

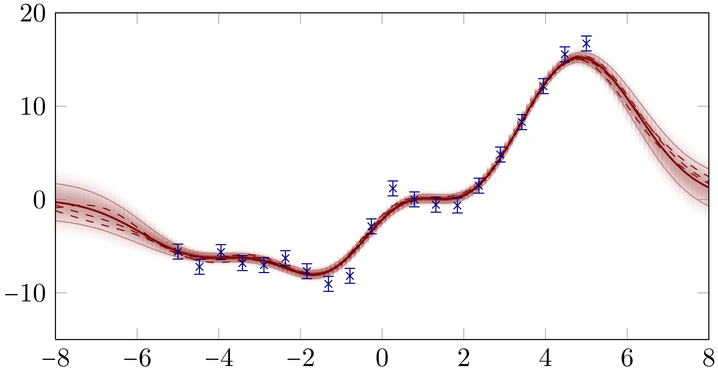


Scaling Outputs

```
k = @(a,b)(1.^2 * exp(-(bsxfun(@minus,a./2,b'./2)).^2));
```

$$v^\top k v \geq 0 \ \forall v \quad \Rightarrow \quad v^\top \theta^2 k v = \theta^2 v^\top k v \geq 0 \ \forall v$$

$$p(f) = \mathcal{GP}(f; \mu, k) \quad \Rightarrow \quad \mathrm{var}[f(x)] = \theta^2 k(x, x)$$

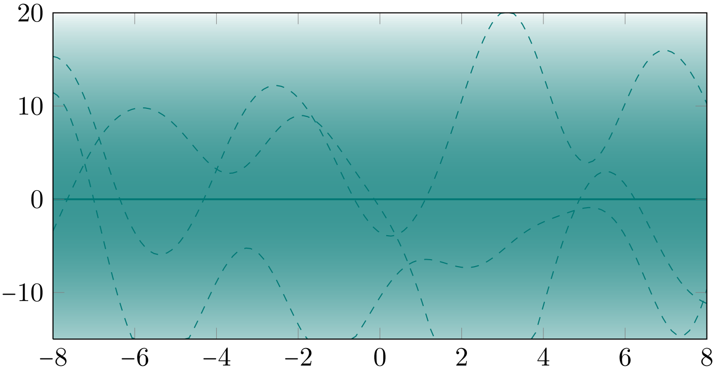


Scaling Outputs

```
k = @(a,b)(10.^2 * exp(-(bsxfun(@minus,a./2,b'./2)).^2));
```

$$v^\top k v \geq 0 \ \forall v \quad \Rightarrow \quad v^\top \theta^2 k v = \theta^2 v^\top k v \geq 0 \ \forall v$$

$$p(f) = \mathcal{GP}(f; \mu, k) \quad \Rightarrow \quad \mathrm{var}[f(x)] = \theta^2 k(x,x)$$

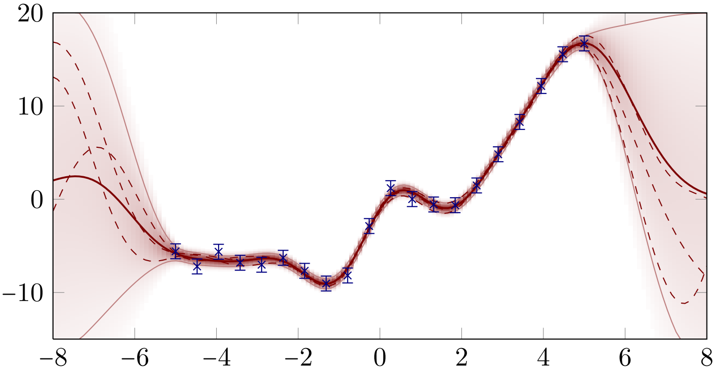


Scaling Outputs

```
k = @(a,b)(10.^2 * exp(-(bsxfun(@minus,a./2,b'./2)).^2));
```

$$v^\top k v \geq 0 \ \forall v \qquad \Rightarrow \qquad v^\top \theta^2 k v = \theta^2 v^\top k v \geq 0 \ \forall v$$

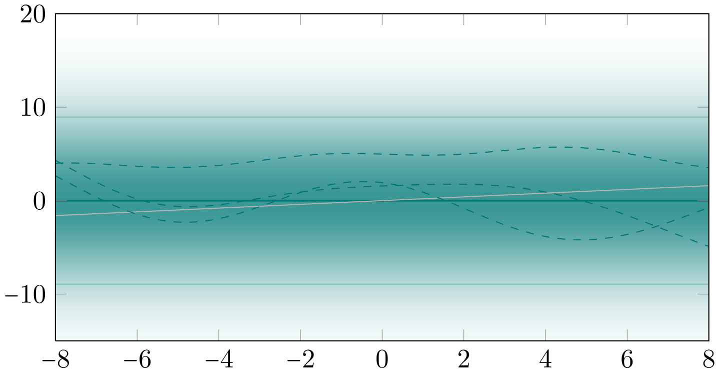
$$p(f) = \mathcal{GP}(f; \mu, k) \qquad \Rightarrow \qquad \mathrm{var}[f(x)] = \theta^2 k(x, x)$$



Scaling Inputs

```
kSE = @(a,b)(exp(-(bsxfun(@minus,a,b')).^2)); phi = @(a)(a/5);
k   = @(a,b)(20 * kSE(phi(a),phi(b)));
```

$$k(a,b) = \sum_\ell \eta_\ell(a)\eta_\ell(b)^\top \qquad \Rightarrow \qquad k(\phi(a),\phi(b)) = \sum_\ell \eta_\ell(\phi(a))\eta_\ell(\phi(b))^\top$$

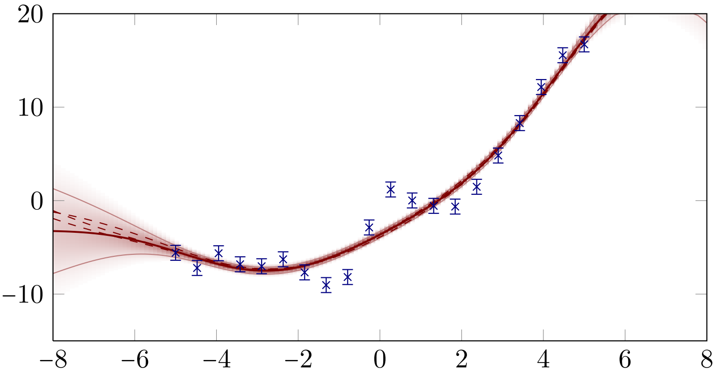


- $k(a,b)$ is pos. semidef. $\Rightarrow$ $k(\phi(a),\phi(b))$ is pos. semidef.

Scaling Inputs

```
kSE = @(a,b)(exp(-(bsxfun(@minus,a,b')).^2)); phi = @(a)(a/5);
k   = @(a,b)(20 * kSE(phi(a),phi(b)));
```

$$k(a,b) = \sum\!\!\!\!\!\!\int_\ell \eta_\ell(a)\eta_\ell(b)^\top \qquad \Rightarrow \qquad k(\phi(a),\phi(b)) = \sum\!\!\!\!\!\!\int_\ell \eta_\ell(\phi(a))\eta_\ell(\phi(b))^\top$$

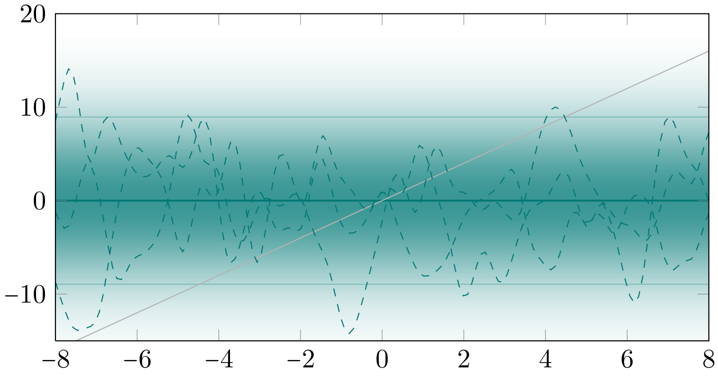


- $k(a,b)$ is pos. semidef. $\Rightarrow k(\phi(a),\phi(b))$ is pos. semidef.

Scaling Inputs

```
kSE = @(a,b)(exp(-(bsxfun(@minus,a,b')).^2)); phi = @(a)(a*2);
k   = @(a,b)(20 * kSE(phi(a),phi(b)));
```

$$k(a,b) = \sum\!\!\!\!\!\!\int_\ell \eta_\ell(a)\eta_\ell(b)^\top \quad \Rightarrow \quad k(\phi(a),\phi(b)) = \sum\!\!\!\!\!\!\int_\ell \eta_\ell(\phi(a))\eta_\ell(\phi(b))^\top$$

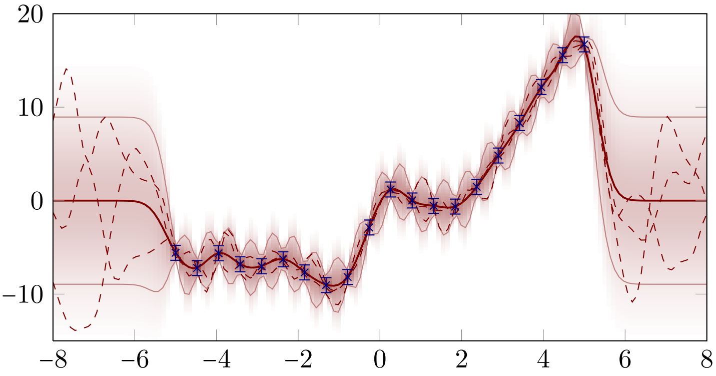


- $k(a,b)$ is pos. semidef. $\Rightarrow k(\phi(a),\phi(b))$ is pos. semidef.

Scaling Inputs

```
kSE = @(a,b)(exp(-(bsxfun(@minus,a,b')).^2)); phi = @(a)(a*2);
k   = @(a,b)(20 * kSE(phi(a),phi(b)));
```

$$k(a,b) = \sum_{\ell} \eta_\ell(a)\eta_\ell(b)^\intercal \qquad \Rightarrow \qquad k(\phi(a),\phi(b)) = \sum_{\ell} \eta_\ell(\phi(a))\eta_\ell(\phi(b))^\intercal$$

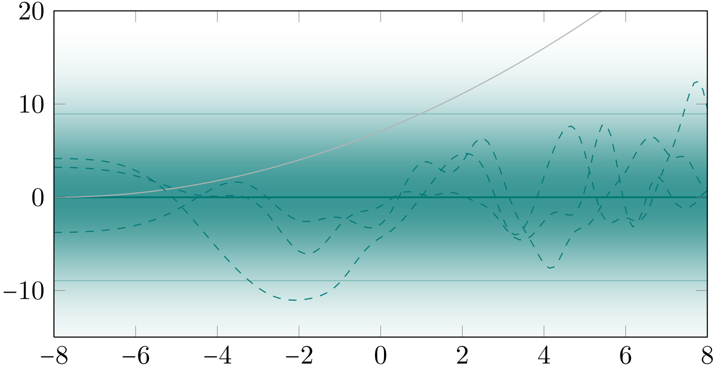


- $k(a,b)$ is pos. semidef. $\Rightarrow$ $k(\phi(a),\phi(b))$ is pos. semidef.

Scaling Inputs

```
kSE = @(a,b)(exp(-(bsxfun(@minus,a,b')).^2)); phi = @(a)(((a+9)./5).^2);
k   = @(a,b)(20 * kSE(phi(a),phi(b)));
```

$$k(a,b) = \sum\!\!\!\!\!\!\int_\ell \eta_\ell(a)\eta_\ell(b)^\intercal \qquad \Rightarrow \qquad k(\phi(a),\phi(b)) = \sum\!\!\!\!\!\!\int_\ell \eta_\ell(\phi(a))\eta_\ell(\phi(b))^\intercal$$

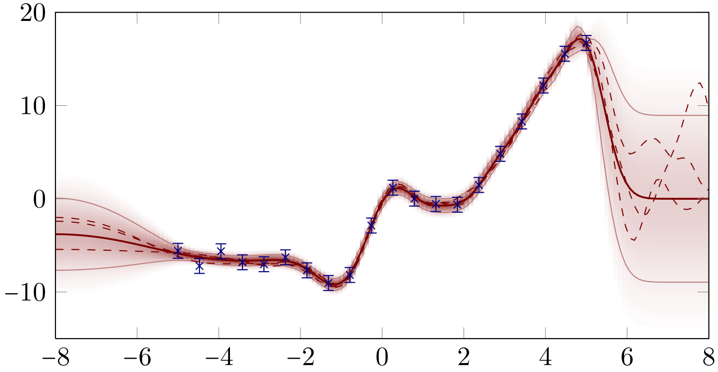


Caution: This can have unintended consequences is ϕ is not monotonic (long range interactions!)

Scaling Inputs

```
kSE = @(a,b)(exp(-(bsxfun(@minus,a,b')).^2)); phi = @(a)(((a+9)./5).^2);
k   = @(a,b)(20 * kSE(phi(a),phi(b)));
```

$$k(a,b) = \sum_{\ell} \eta_\ell(a)\eta_\ell(b)^\top \qquad \Rightarrow \qquad k(\phi(a),\phi(b)) = \sum_{\ell} \eta_\ell(\phi(a))\eta_\ell(\phi(b))^\top$$

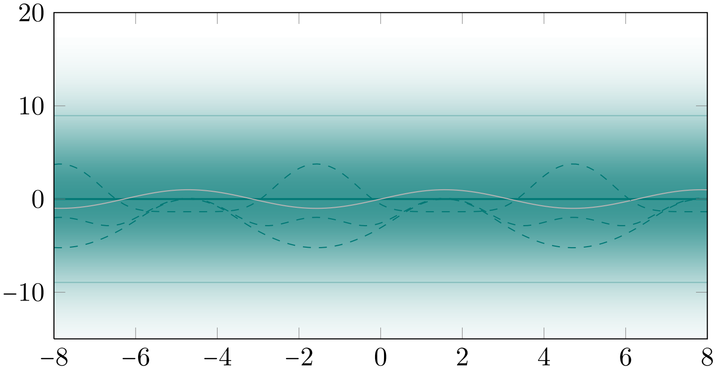


Caution: This can have unintended consequences is ϕ is not monotonic (long range interactions!)

Scaling Inputs – Example: periodic functions

D.J.C. MacKay, 1998

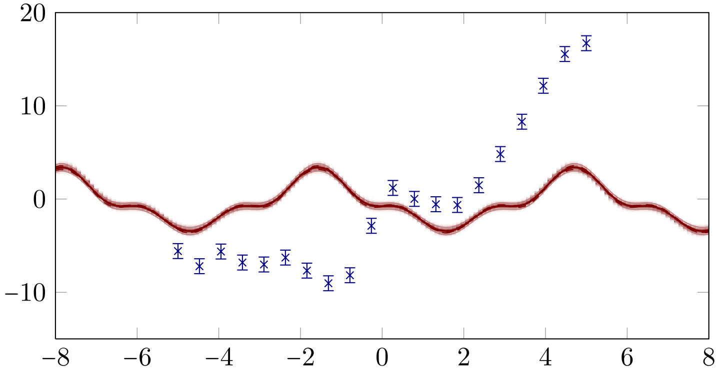
```
phi = @(a)(sin(a)); kSE = @(a,b)(20 * exp(-(bsxfun(@minus,a./2,b'./2)).^2));
k   = @(a,b)(kSE(phi(a),phi(b)));
```



Scaling Inputs – Example: periodic functions

D.J.C. MacKay, 1998

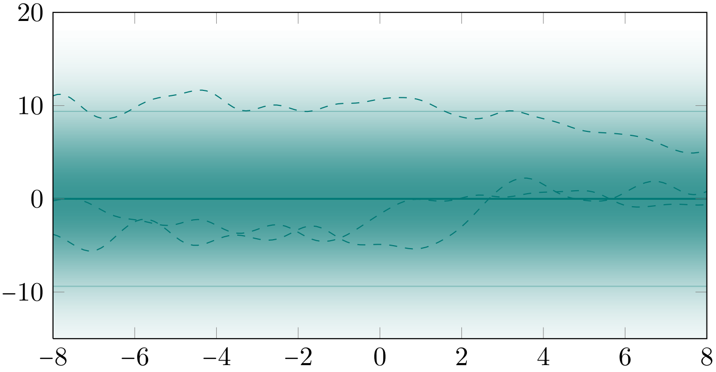
```
phi = @(a)(sin(a)); kSE = @(a,b)(20 * exp(-(bsxfun(@minus,a./2,b'./2)).^2));
k   = @(a,b)(kSE(phi(a),phi(b)));
```



Sums of Kernels are Kernels

```
k1 = @(a,b)(4.^2 * exp(-(bsxfun(@minus,a./2,b'./2)).^2 ./ 10.^2));
k2 = @(a,b)(1.^2 * exp(-(bsxfun(@minus,a./2,b'./2)).^2 ./ 0.5^2));
k  = @(a,b)(k1(a,b) + k2(a,b));
```

$$v^\top(k^1_{XX} + k^2_{XX})v = v^\top k^1_{XX} v + v^\top k^2_{XX} v \geq 0$$

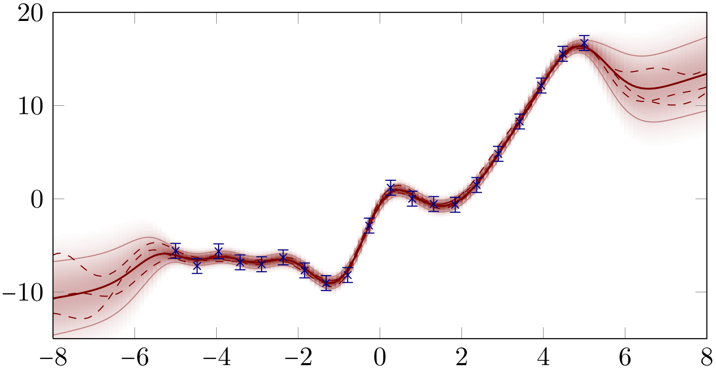


Intuition: similarity under k^1 OR k^2.

Sums of Kernels are Kernels

```
k1 = @(a,b)(4.^2 * exp(-(bsxfun(@minus,a./2,b'./2)).^2 ./ 10.^2));
k2 = @(a,b)(1.^2 * exp(-(bsxfun(@minus,a./2,b'./2)).^2 ./ 0.5^2));
k  = @(a,b)(k1(a,b) + k2(a,b));
```

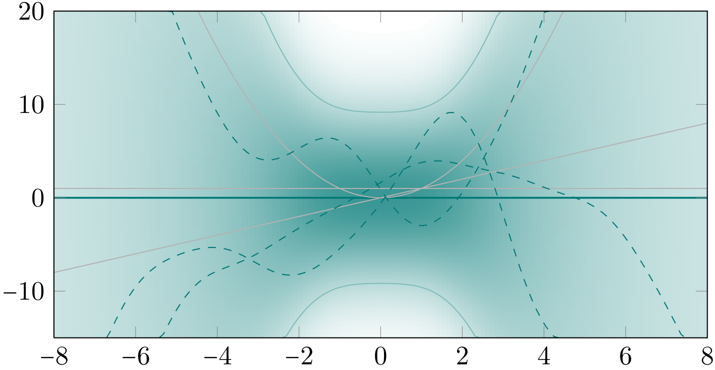
$$v^\top (k^1_{XX} + k^2_{XX})v = v^\top k^1_{XX} v + v^\top k^2_{XX} v \geq 0$$



Intuition: similarity under k^1 OR k^2.

Sums of Kernel and Parametric Features

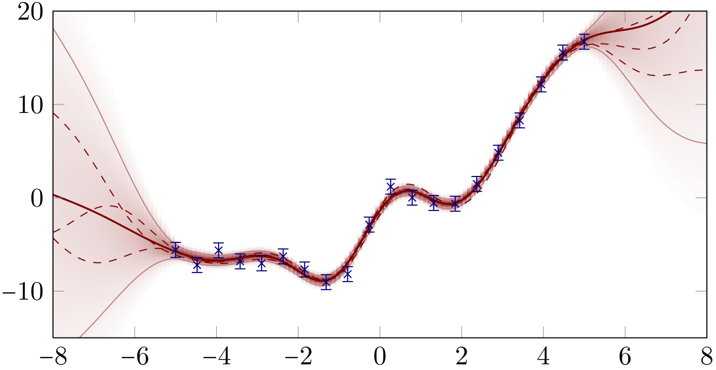
```
phi = @(a)(bsxfun(@power,a,[0:2]));
k   = @(a,b)(20 * exp(-(bsxfun(@minus,a./2,b'./2)).^2) + phi(a)*phi(b)');
```



see Rasmussen & Williams, §2.7 for an efficient implementation

Sums of Kernel and Parametric Features

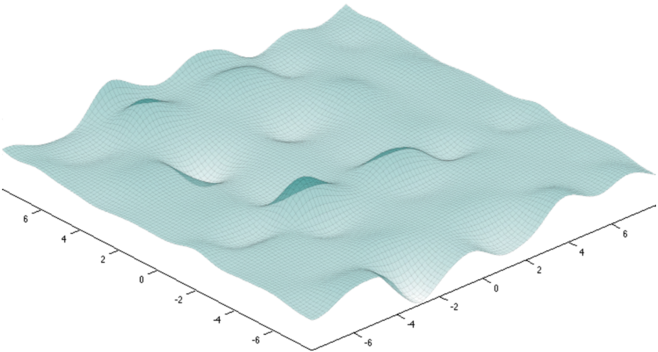
```
phi = @(a)(bsxfun(@power,a,[0:2]));
k   = @(a,b)(20 * exp(-(bsxfun(@minus,a./2,b'./2)).^2) + phi(a)*phi(b)');
```



see Rasmussen & Williams, §2.7 for an efficient implementation

Multiple Inputs

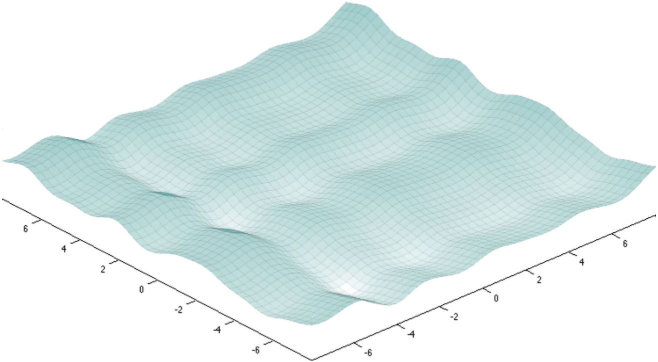
just a quick reminder



Additive Models

```
k = @(a,b)(kSE(a(:,1),b(:,1)) + kSE(a(:,2),b(:,2)));
```

Hastie & Tibshirani, 1990

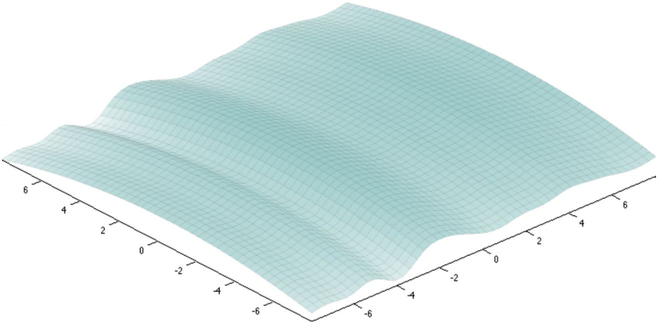


$$k(a,b) = \sum_{d}^{D} k_d(a_d, b_d)$$

Additive Models

```
phi = @(a)(bsxfun(@power,a,[0:2]));
k   = @(a,b)(kSE(a(:,1),b(:,1)) + phi(a(:,2))*phi(b(:,2))');
```

Wahba, 1990, Rasmussen & Williams, 2006



$$k(a,b) = \sum_{d}^{D} k_d(a_d, b_d)$$

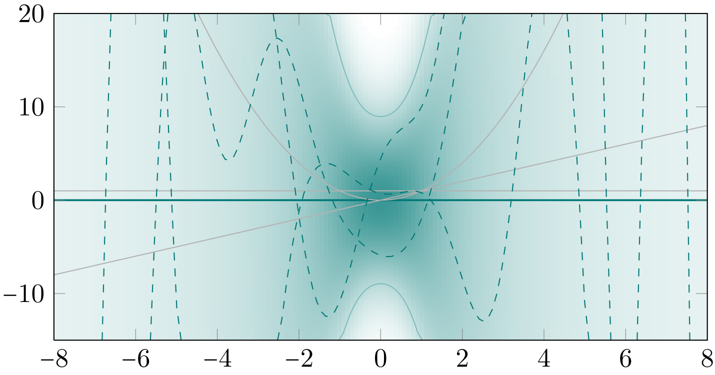
- use structure of k_{XX} to drastically lower inference cost
- generalize to $k(a,b) = \sum_d^D k_d(a_d, b_d) + \sum_i^D \sum_j^{i-1} k_{ij}(a_i, a_j, b_i, b_j)$ to get functional ANOVA

Products of Kernels are Kernels

```
phi = @(a)(bsxfun(@power,a,[0:2]));
k1  = @(a,b)(20 * exp(-(bsxfun(@minus,a./2,b'./2)).^2));
k   = @(a,b)(k1(a,b) .* (phi(a) * phi(b)'));
```

Theorem (I. Schur (proof in Bapat, 1997, Million 2007))

If A and B are positive semidefinite, then $A \odot B$ (=`A.*B`) is semidefinite.



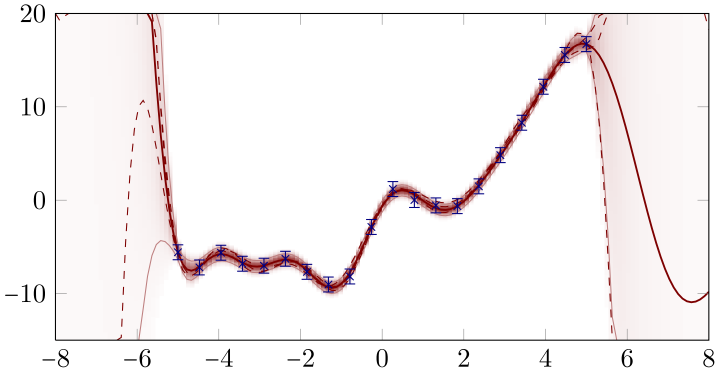
Intuition: similarity under k^1 AND k^2.

Products of Kernels are Kernels

```
phi = @(a)(bsxfun(@power,a,[0:2]));
k1  = @(a,b)(20 * exp(-(bsxfun(@minus,a./2,b'./2)).^2));
k   = @(a,b)(k1(a,b) .* (phi(a) * phi(b)'));
```

Theorem (I. Schur (proof in Bapat, 1997, Million 2007))

If A and B are positive semidefinite, then $A \odot B$ (=`A.*B`) is semidefinite.



Intuition: similarity under k^1 AND k^2.

Summary: Kernel design

Mercer kernels form a semiring

- k is positive semidefinite $\Rightarrow \alpha k$ for $\alpha \in \mathbb{R}_+$ is positive semidefinite
  e.g. to change signal variance
- $k(a, b)$ is pos. semidef. $\Rightarrow k(\phi(a), \phi(b))$ is pos. semidef.
  e.g. to change length scale
- k_1, k_2 is positive semidefinite $\Rightarrow k_1 + k_2$ is positive semidefinite
  e.g. to encode OR similarity
- k_1, k_2 is positive semidefinite $\Rightarrow k_1 \odot k_2$ is positive semidefinite
  e.g. to encode AND similarity

These rules can encode prior knowledge in Gaussian models.

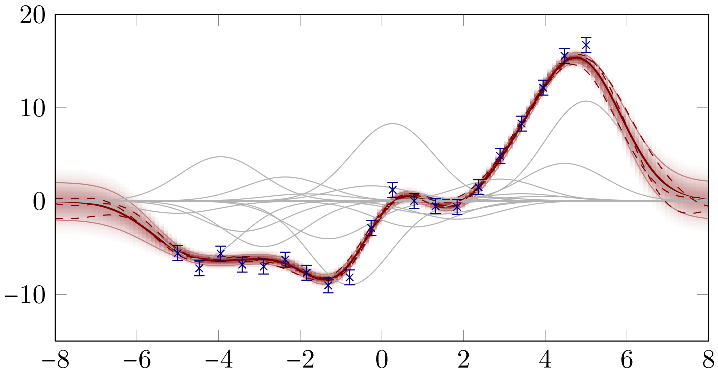
If your model has no parameters, you haven't found them yet.

- Gaussians link *inference* and *linear algebra*
- linear weights with features model *functions*
- in fact, the number of features can be infinite!
- kernels can be
  - *output scaled*
  - *input scaled*
  - *added*
  - *multiplied*

  to get more expressive models
- but every kernel remains a nontrivial assumption

Reproducing Kernel Hilbert Spaces

the very rough story



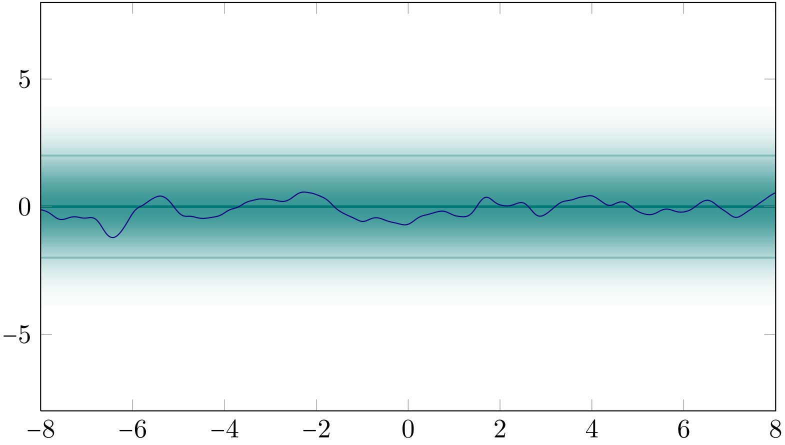
- posterior mean $k_{xX}(k_{XX} + \sigma^2 I)^{-1} y = k_{xX}\alpha$
- so we are interested in the space of functions (the RKHS)

$$f(x) = \sum_i^N \alpha_i k(x, X_i) \qquad \text{for various } X_i,\ N,\ \alpha.$$

- for some kernels (SE, RQ, OU, ...), this space lies dense in the

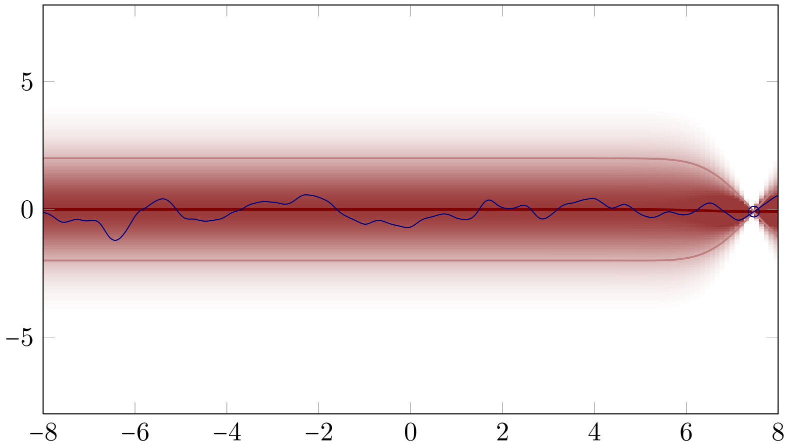
Universal RKHSs

an experiment – prior



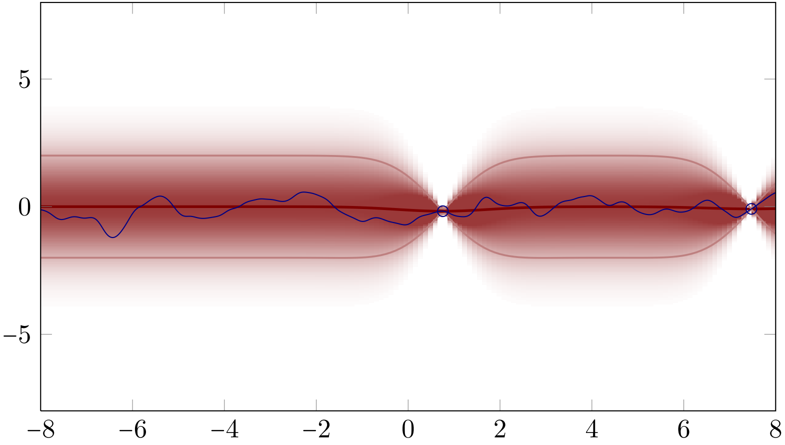
Universal RKHSs

an experiment – 1 evaluation



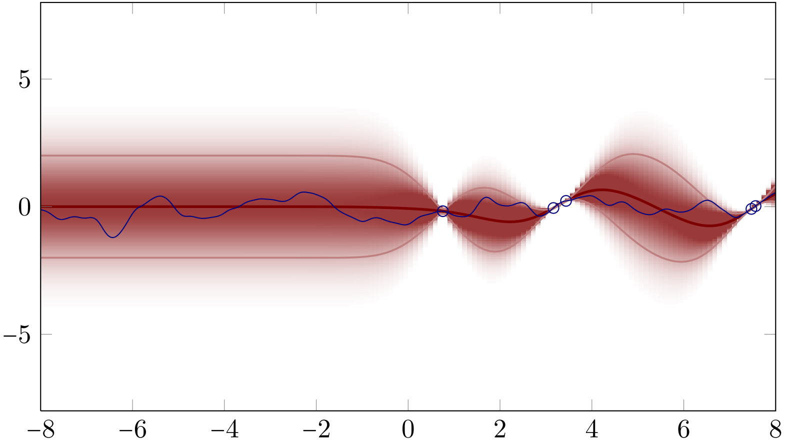
Universal RKHSs

an experiment – 2 evaluations



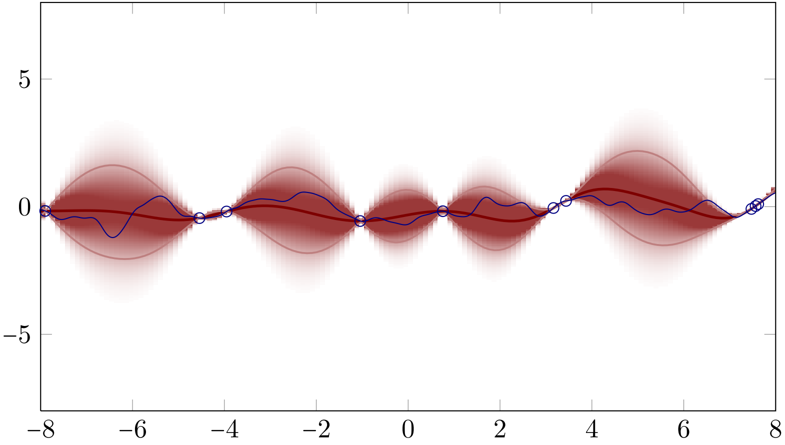
Universal RKHSs

an experiment – 5 evaluations



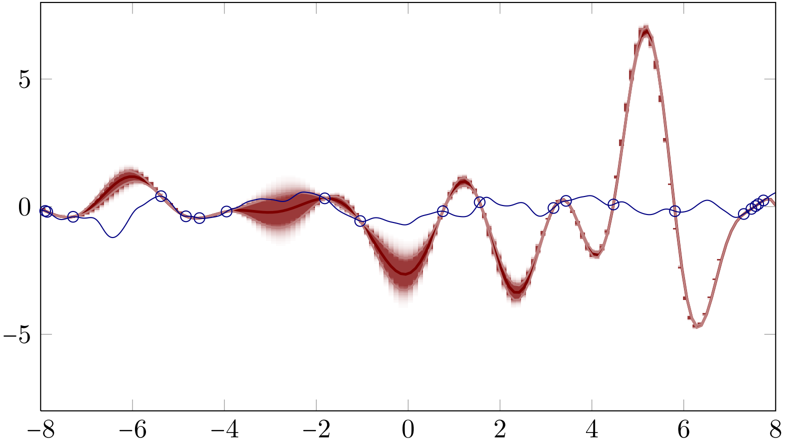
Universal RKHSs

an experiment – 10 evaluations



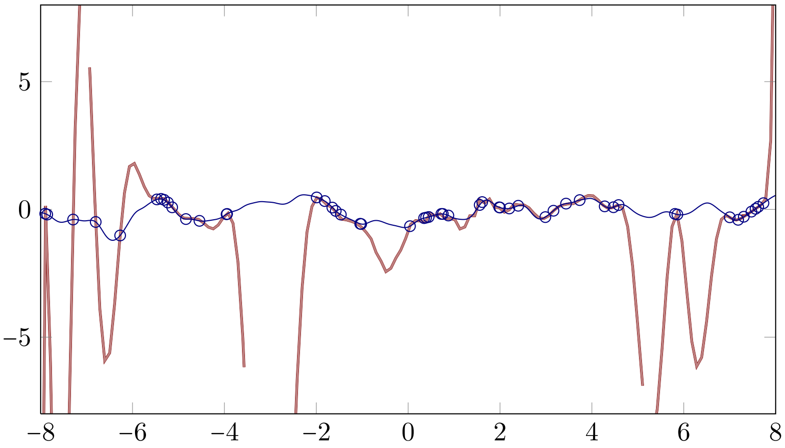
Universal RKHSs

an experiment – 20 evaluations



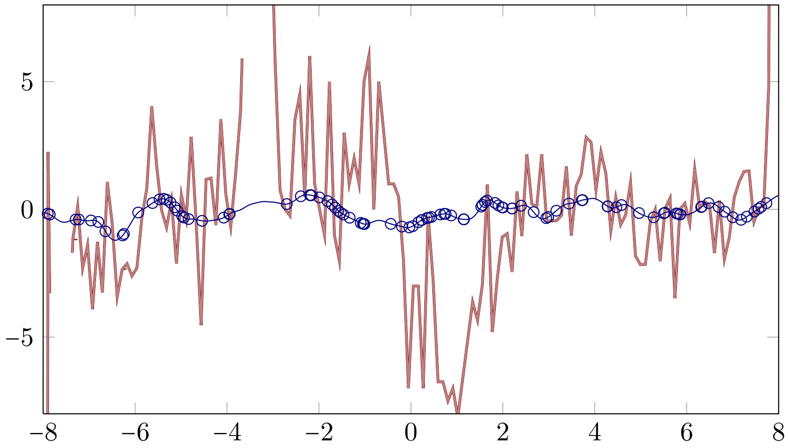
Universal RKHSs

an experiment – 50 evaluations



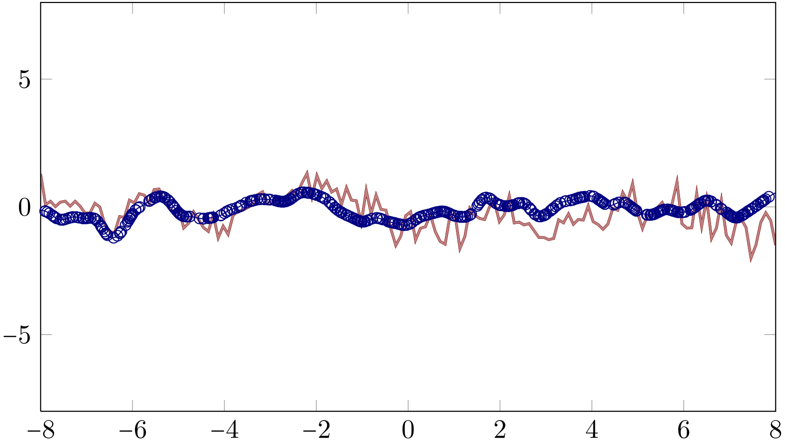
Universal RKHSs

an experiment – 100 evaluations



Universal RKHSs

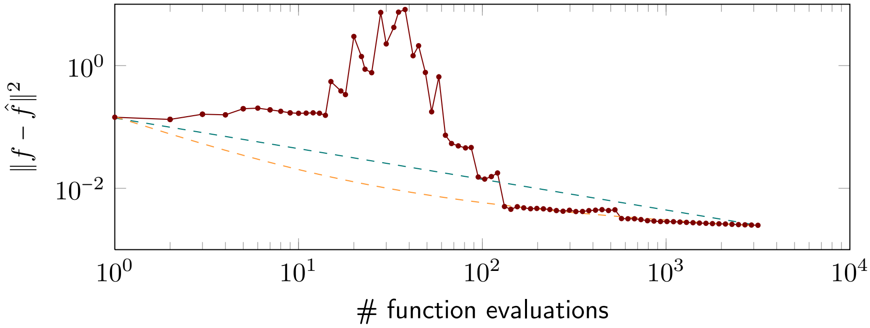
an experiment – 500 evaluations



Convergence Rates are Important

non-obvious aspects of f can ruin convergence

v.d.Vaart & v.Zanten, 2011



If f is "not well represented" by the kernel (has low prior density), the number of datapoints required to achieve ϵ error can be exponential in ϵ. Outside of the observation range, there are no guarantees at all.

v.d.Vaart & v.Zanten. *Information Rates of Nonparametric GP models*. JMLR 12 (2011)

An Analogy

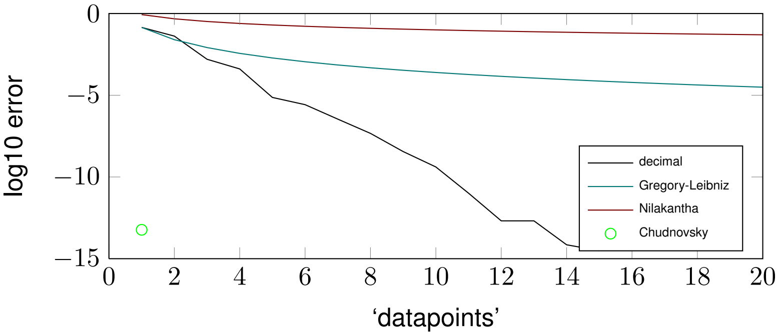
representing π in $\mathbb{Q}$

- $\mathbb{Q}$ is dense in $\mathbb{R}$

$$\pi = 3 \cdot \frac{1}{1} + 1 \cdot \frac{1}{10} + 4 \cdot \frac{1}{100} + 1 \cdot \frac{1}{1000} + \ldots \qquad \text{decimal}$$

$$= 4 \cdot \frac{1}{1} - 4 \cdot \frac{1}{3} + 4 \cdot \frac{1}{5} - 4 \cdot \frac{1}{7} + \ldots \qquad \text{Gregory-Leibniz}$$

$$= 3 \cdot \frac{1}{1} + 4 \cdot \frac{1}{2 \cdot 3 \cdot 4} - 4 \cdot \frac{1}{4 \cdot 5 \cdot 6} + 4 \cdot \frac{1}{6 \cdot 7 \cdot 8} \qquad \text{Nilakantha}$$



Summary

- Gaussians link inference and linear algebra
- linear weights with features model functions
- in fact, number of features can be infinite $\rightarrow$ GP regression
- kernels can be
  - output scaled
  - input scaled
  - added
  - multiplied

  to get more expressive models
- but every kernel remains a nontrivial assumption

GPs with universal kernels can learn every continuous function!
But they learn some functions exponentially slower than others.

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