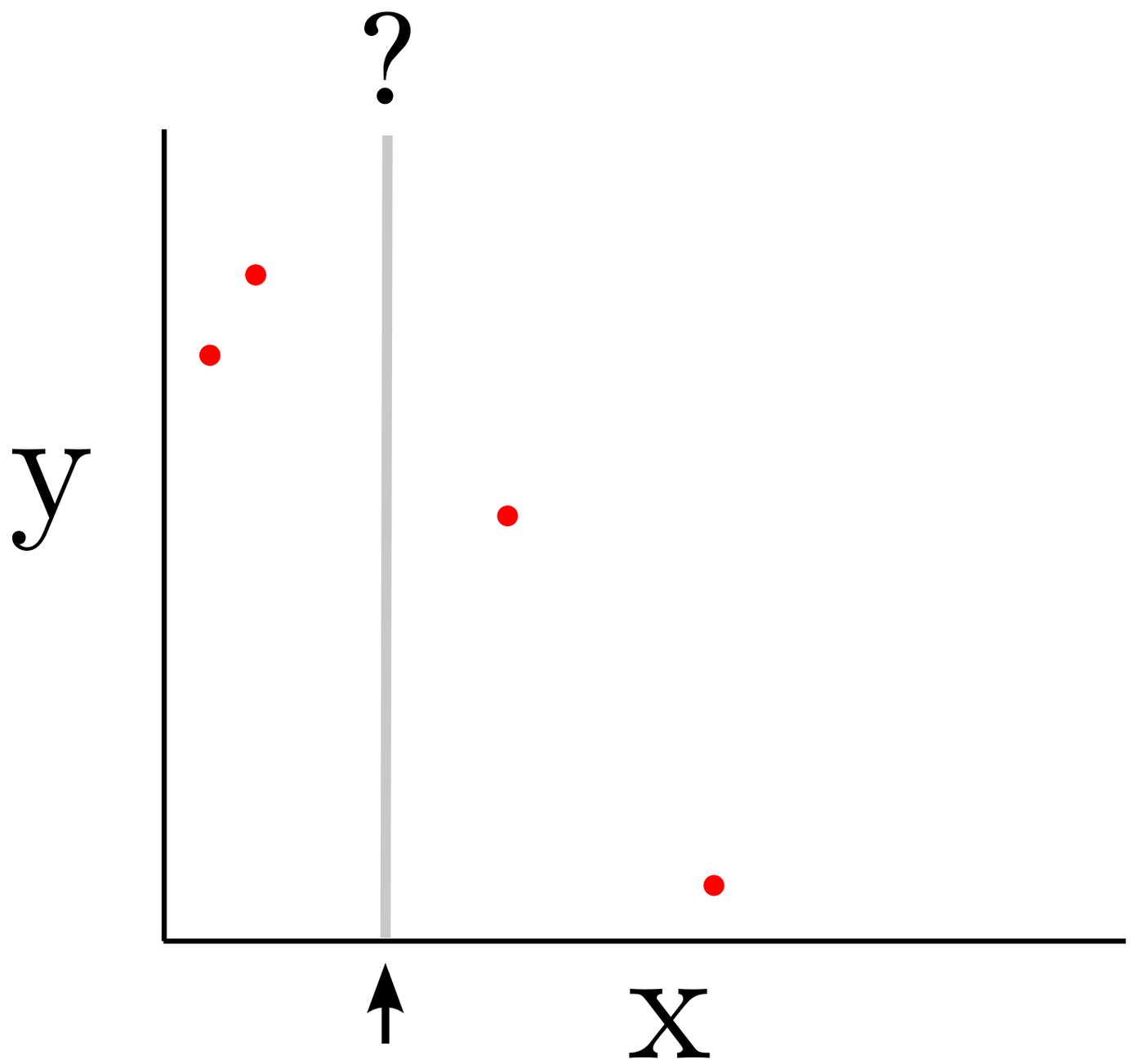
DS-GA 3001.008 Modeling time series data

L11. GPs

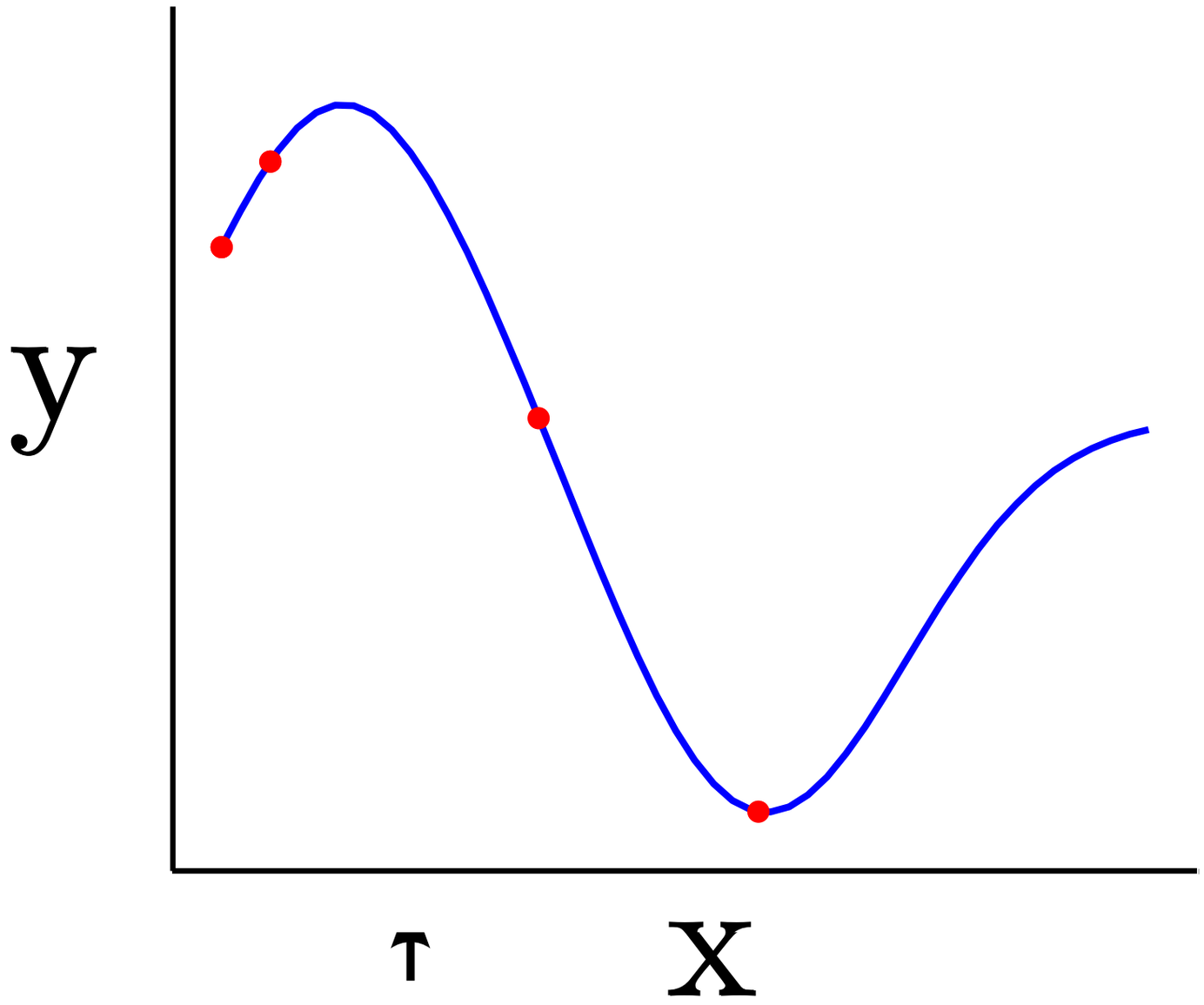
Instructor: Cristina Savin

NYU, CNS & CDS

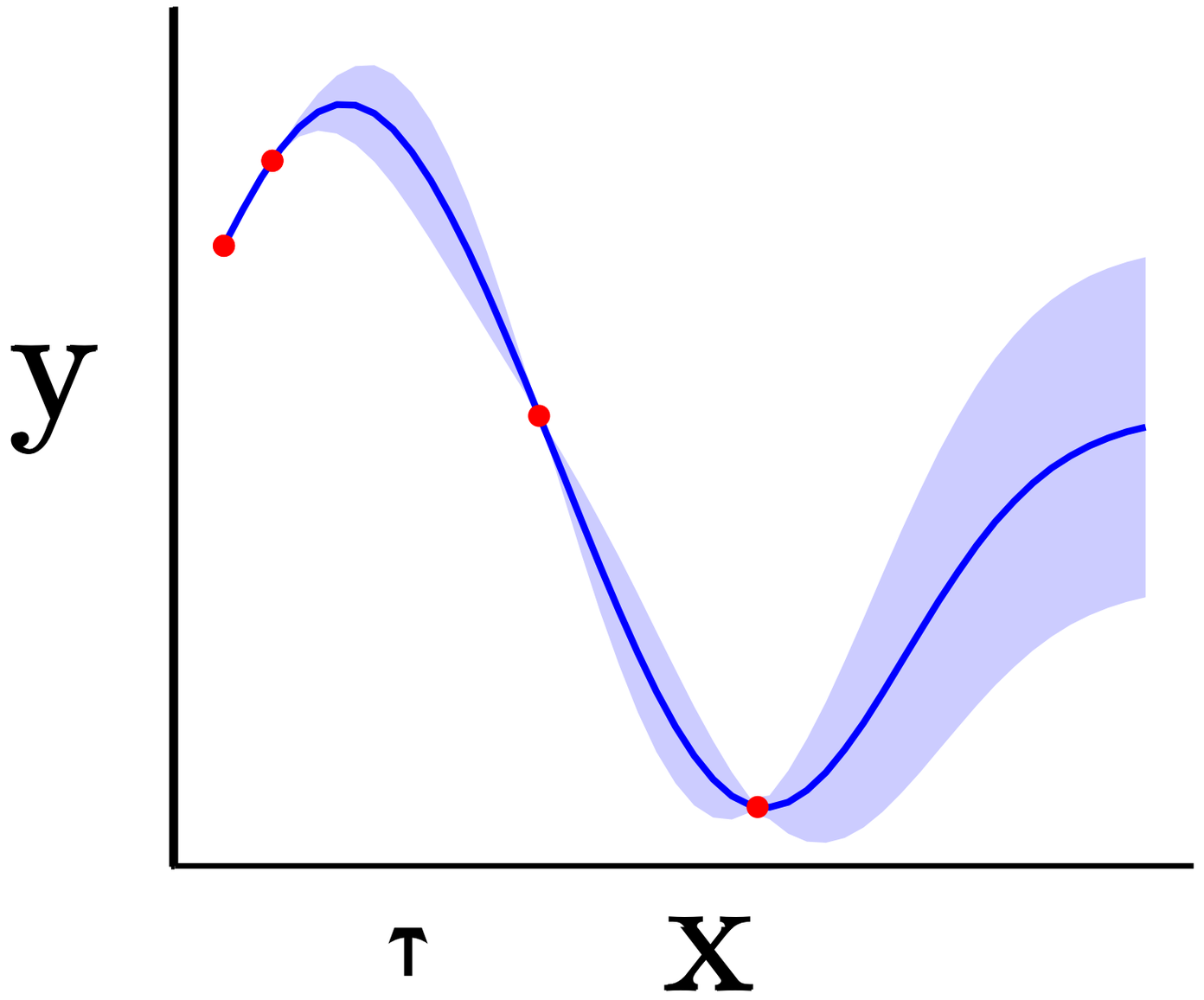
Quick recap
1. Motivation: nonlinear regression



Quick recap
1. Motivation: nonlinear regression



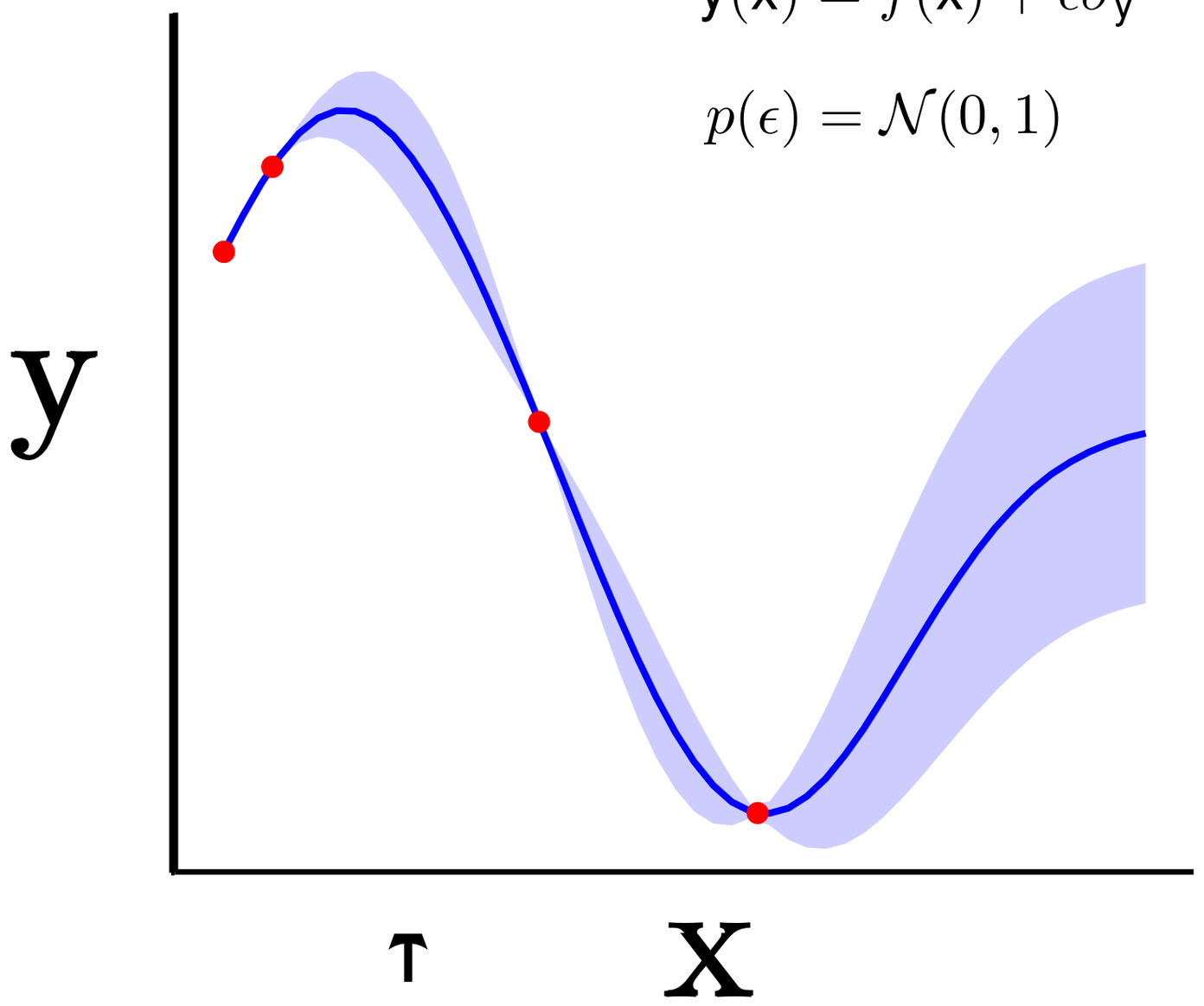
Quick recap
1. Motivation: nonlinear regression



Quick recap
1. Motivation: nonlinear regression

$$\mathsf{y}(\mathsf{x}) = f(\mathsf{x}) + \epsilon\sigma_{\mathsf{y}}$$

$$p(\epsilon) = \mathcal{N}(0, 1)$$

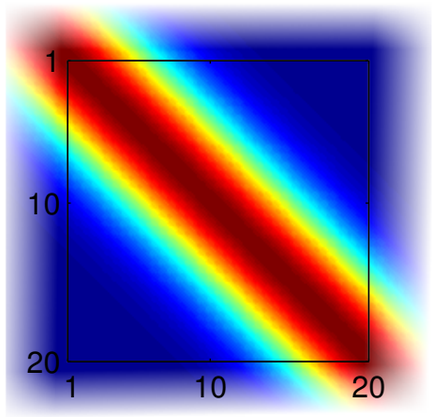


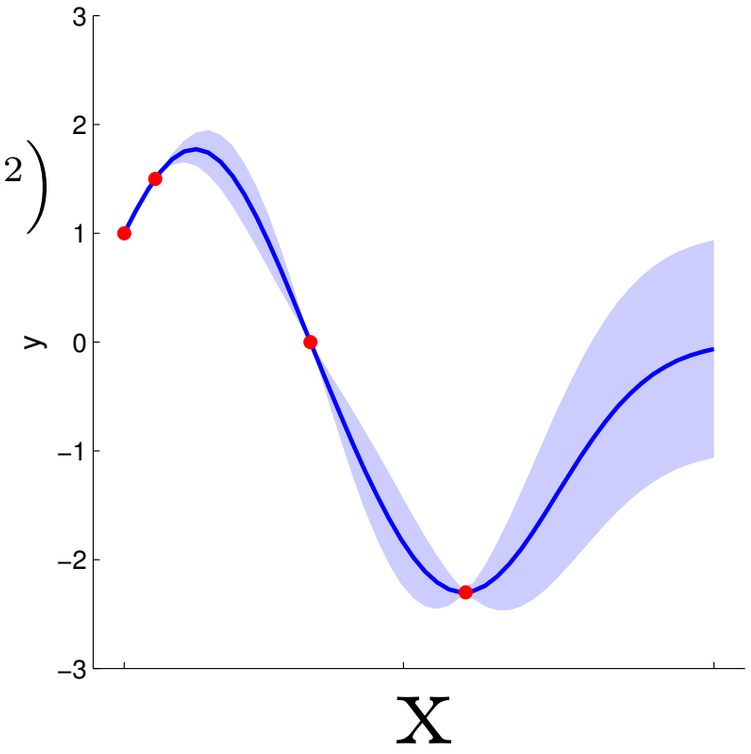
Quick recap
2.prior over functions: GP

$$\Sigma(\mathsf{x}_1, \mathsf{x}_2) = \mathsf{K}(\mathsf{x}_1, \mathsf{x}_2) + \mathsf{I}\sigma_{\mathsf{y}}^2$$

$$\mathsf{K}(\mathsf{x}_1, \mathsf{x}_2) = \sigma^2 \exp\left(-\frac{1}{2\mathsf{l}^2}(\mathsf{x}_1 - \mathsf{x}_2)^2\right)$$

$\Sigma =$





GP: generalization of multivariate gaussian

Definition: a Gaussian process is a collection of random variables, any finite number of which have (consistent) Gaussian distributions.

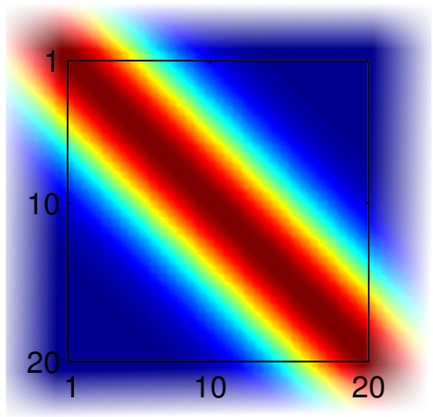
$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), \mathsf{K}(\mathbf{x}, \mathbf{x}')\right)$ mean+covariance functions

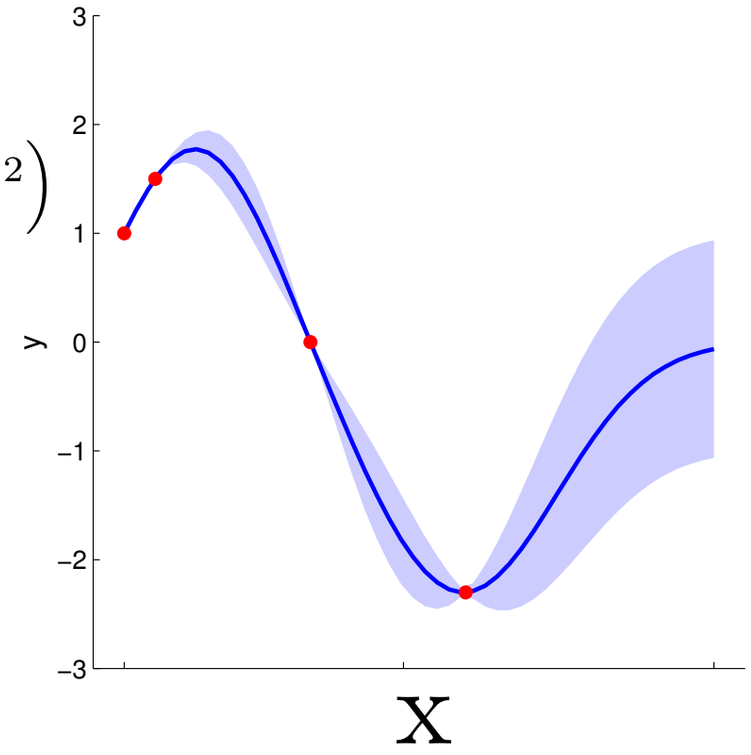
Quick recap
2.prior over functions: GP

$$\Sigma(\mathsf{x}_1, \mathsf{x}_2) = \mathsf{K}(\mathsf{x}_1, \mathsf{x}_2) + \mathsf{I}\sigma_{\mathsf{y}}^2$$

$$\mathsf{K}(\mathsf{x}_1, \mathsf{x}_2) = \sigma^2 \exp\left(-\frac{1}{2\mathsf{l}^2}(\mathsf{x}_1 - \mathsf{x}_2)^2\right)$$

$\Sigma =$





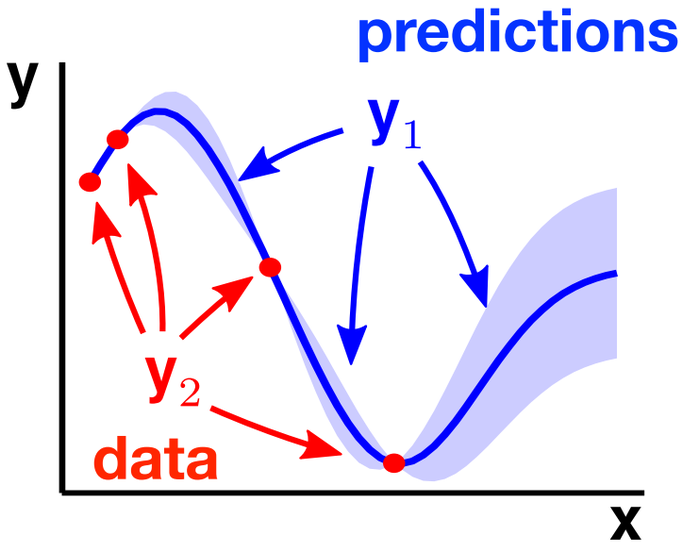
GP: generalization of multivariate gaussian

Definition: a Gaussian process is a collection of random variables, any finite number of which have (consistent) Gaussian distributions.

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), \mathsf{K}(\mathbf{x}, \mathbf{x}')\right)$$ mean+covariance functions

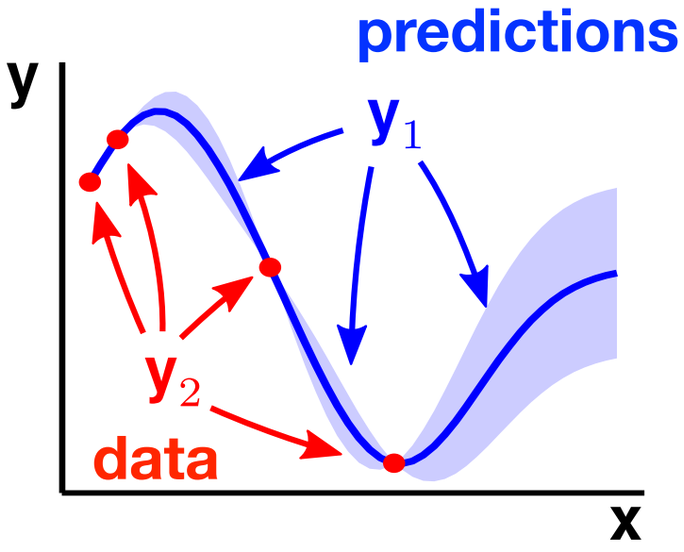
Any function has nonzero probability, preference for certain structure

Quick recap
3. GP inference



$$p(\mathbf{y}_1|\mathbf{y}_2) = \frac{p(\mathbf{y}_1, \mathbf{y}_2)}{p(\mathbf{y}_2)}$$

Quick recap
3. GP inference

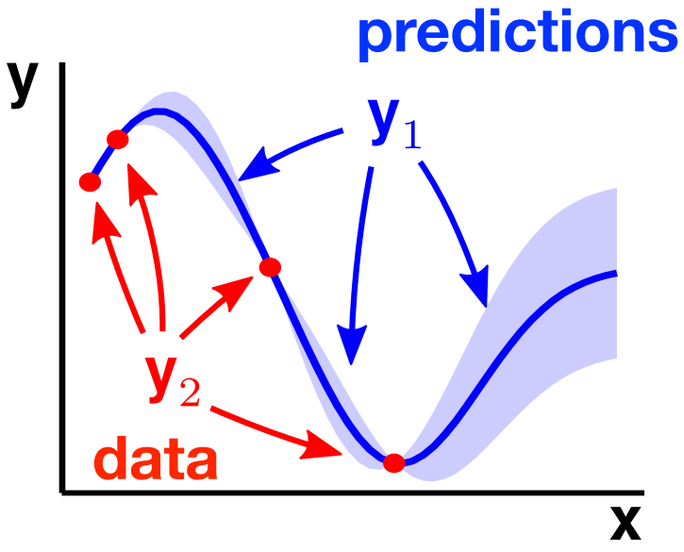


$$p(\mathbf{y}_1|\mathbf{y}_2) = \frac{p(\mathbf{y}_1, \mathbf{y}_2)}{p(\mathbf{y}_2)}$$

Jointly gaussian:

$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{B}^\mathsf{T} & \mathsf{C} \end{bmatrix}\right)$$

Quick recap
3. GP inference



$$p(\mathbf{y}_1|\mathbf{y}_2) = \frac{p(\mathbf{y}_1, \mathbf{y}_2)}{p(\mathbf{y}_2)}$$

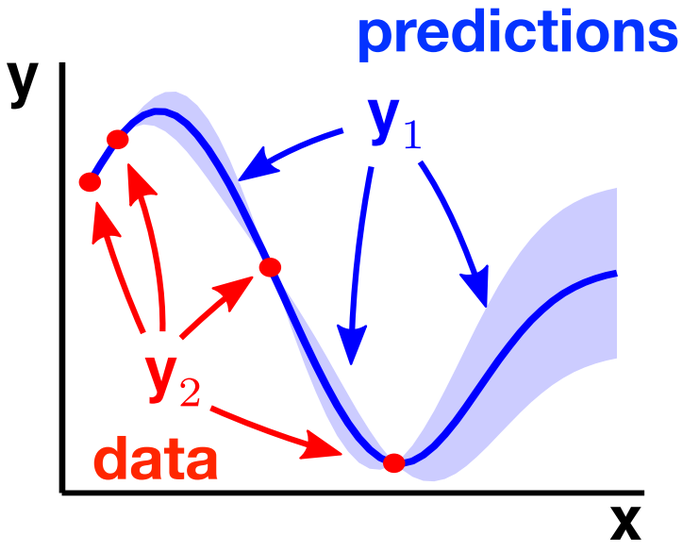
Jointly gaussian:

$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{B}^\intercal & \mathsf{C} \end{bmatrix}\right)$$

$$\downarrow$$

$$p(\mathbf{y}_1|\mathbf{y}_2) = \mathcal{N}(\mathbf{a} + \mathsf{B}\mathsf{C}^{-1}(\mathbf{y}_2 - \mathbf{b}), \mathsf{A} - \mathsf{B}\mathsf{C}^{-1}\mathsf{B}^\intercal)$$

Quick recap
3. GP inference



$$p(\mathbf{y}_1|\mathbf{y}_2) = \frac{p(\mathbf{y}_1, \mathbf{y}_2)}{p(\mathbf{y}_2)}$$

Jointly gaussian:

$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{B}^\intercal & \mathsf{C} \end{bmatrix}\right)$$

$$\downarrow$$

$$p(\mathbf{y}_1|\mathbf{y}_2) = \mathcal{N}(\mathbf{a} + \mathsf{B}\mathsf{C}^{-1}(\mathbf{y}_2 - \mathbf{b}), \mathsf{A} - \mathsf{B}\mathsf{C}^{-1}\mathsf{B}^\intercal)$$

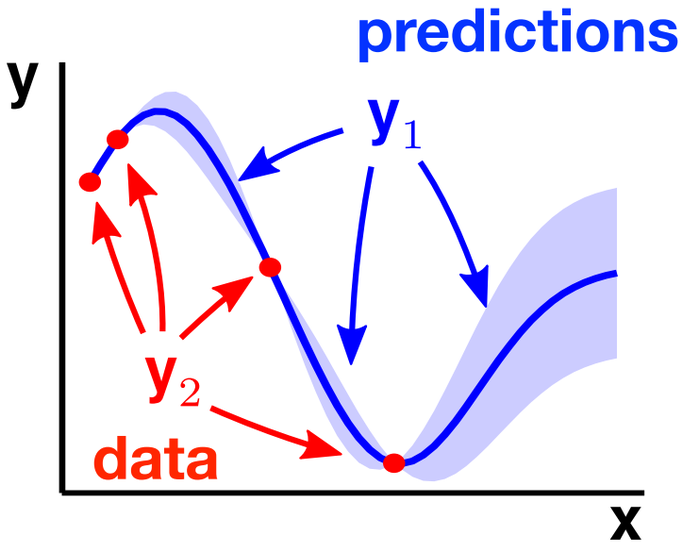
predicted mean

$$\begin{aligned} \mu_{\mathbf{y}_1|\mathbf{y}_2} &= \mathbf{a} + \mathsf{B}\mathsf{C}^{-1}(\mathbf{y}_2 - \mathbf{b}) \\ &= \mathsf{B}\mathsf{C}^{-1}\mathbf{y}_2 \\ &= \mathsf{W}\mathbf{y}_2 \end{aligned}$$

predicted covariance

$$\Sigma_{\mathbf{y}_1|\mathbf{y}_2} = \mathsf{A} - \mathsf{B}\mathsf{C}^{-1}\mathsf{B}^\intercal$$

Quick recap
3. GP inference



$$p(\mathbf{y}_1|\mathbf{y}_2) = \frac{p(\mathbf{y}_1, \mathbf{y}_2)}{p(\mathbf{y}_2)}$$

Jointly gaussian:

$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{B}^\intercal & \mathsf{C} \end{bmatrix}\right)$$

$$\downarrow$$

$$p(\mathbf{y}_1|\mathbf{y}_2) = \mathcal{N}(\mathbf{a} + \mathsf{B}\mathsf{C}^{-1}(\mathbf{y}_2 - \mathbf{b}), \mathsf{A} - \mathsf{B}\mathsf{C}^{-1}\mathsf{B}^\intercal)$$

predicted mean

$$\begin{aligned} \mu_{\mathbf{y}_1|\mathbf{y}_2} &= \mathbf{a} + \mathsf{B}\mathsf{C}^{-1}(\mathbf{y}_2 - \mathbf{b}) \\ &= \mathsf{B}\mathsf{C}^{-1}\mathbf{y}_2 \\ &= \mathsf{W}\mathbf{y}_2 \end{aligned}$$

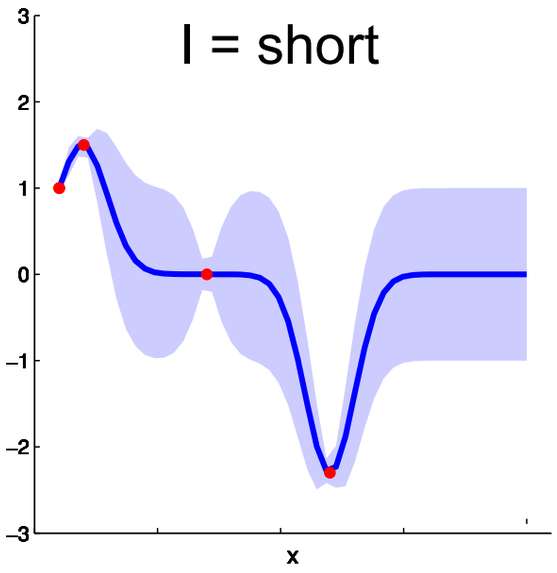
predicted covariance

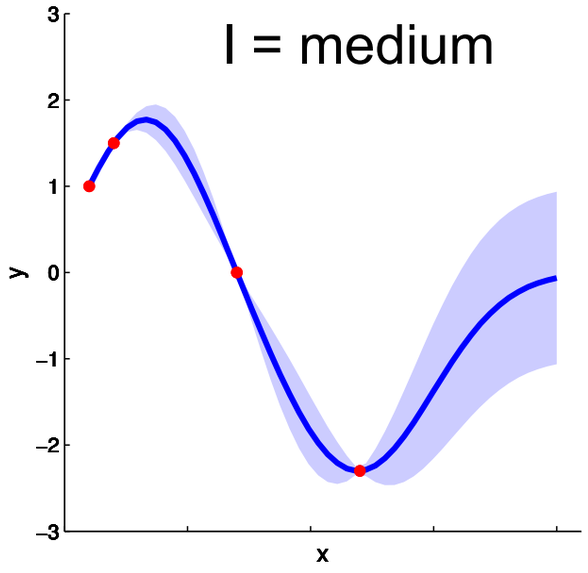
$$\Sigma_{\mathbf{y}_1|\mathbf{y}_2} = \mathsf{A} - \mathsf{B}\mathsf{C}^{-1}\mathsf{B}^\intercal$$

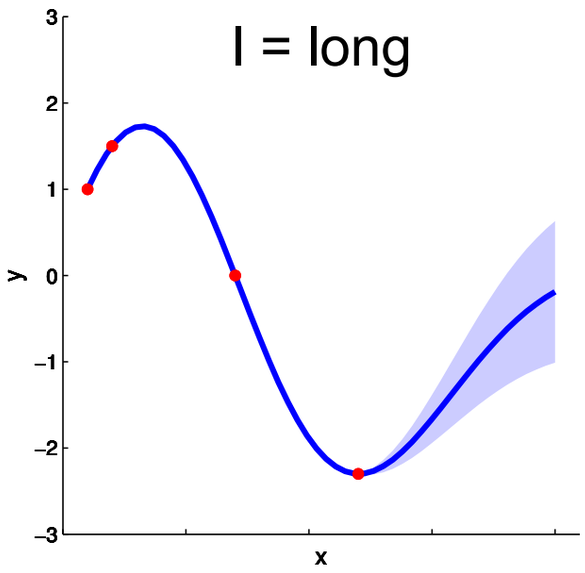
Computational bottleneck!!!

Quick recap
4. GP hyperparameters

$$K(x_1, x_2) = \sigma^2 \exp\left(-\frac{1}{2l^2}(x_1 - x_2)^2\right)$$



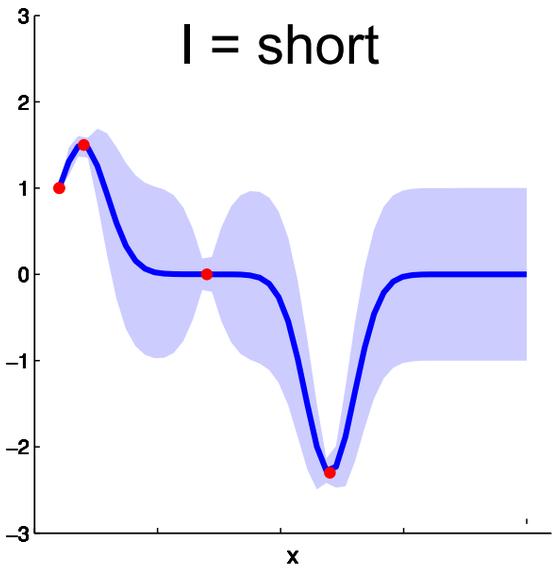


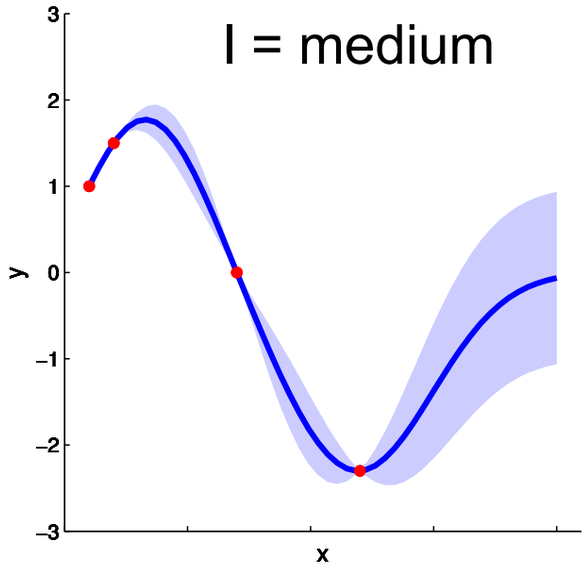


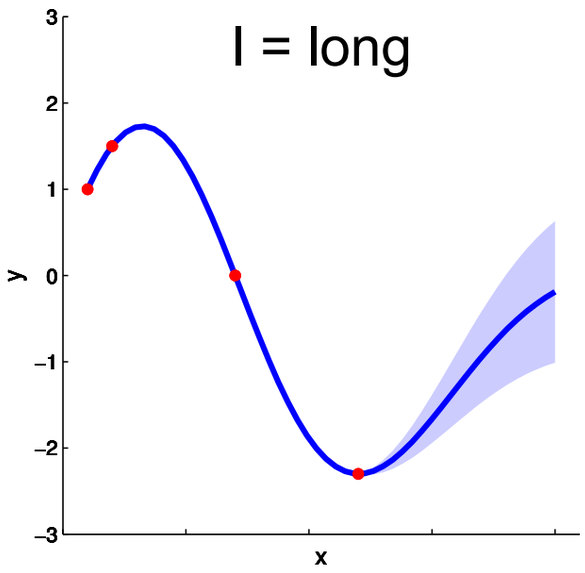
Hyperparameters significantly influence outcome

Quick recap
4. GP hyperparameters

$$K(x_1, x_2) = \sigma^2 \exp\left(-\frac{1}{2l^2}(x_1 - x_2)^2\right)$$





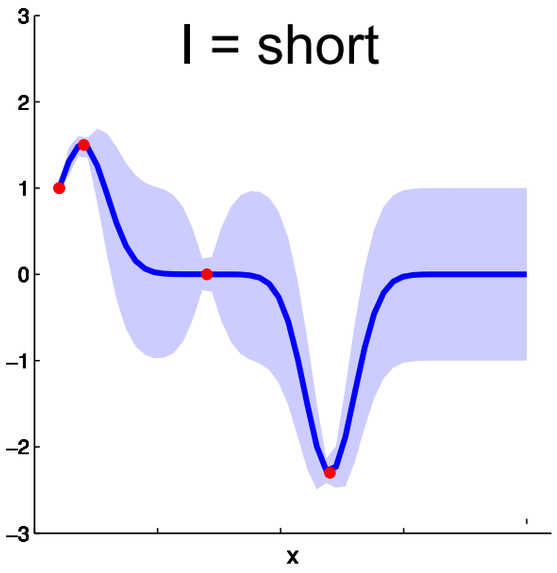


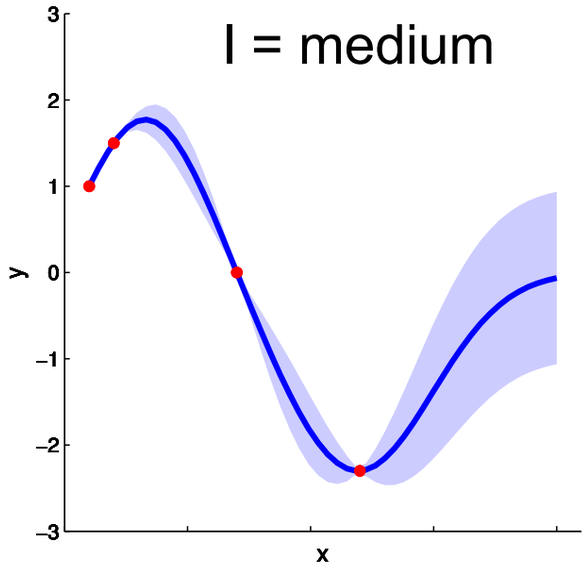
Hyperparameters significantly influence outcome

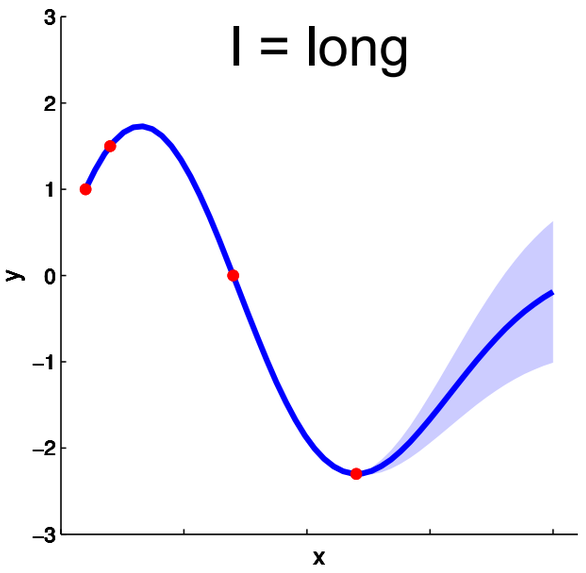
Learn values from data!

Quick recap
4. GP hyperparameters

$$K(x_1, x_2) = \sigma^2 \exp\left(-\frac{1}{2l^2}(x_1 - x_2)^2\right)$$







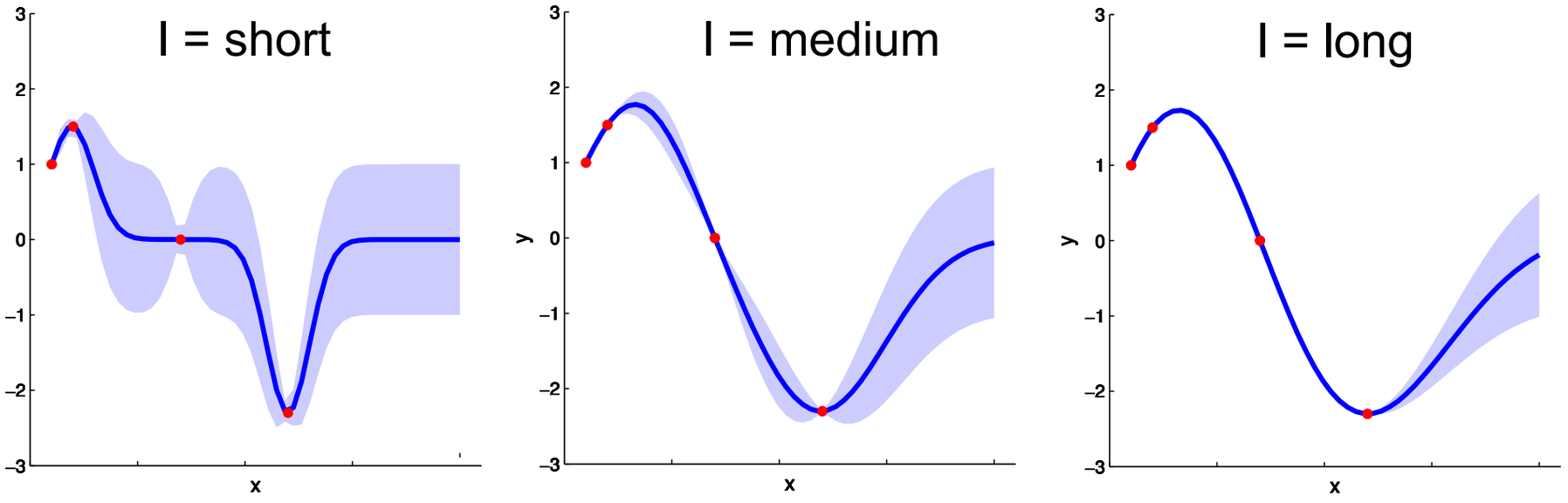
Hyperparameters significantly influence outcome

Learn values from data!

A. Maximum likelihood fit:

Quick recap
4. GP hyperparameters

$$K(x_1, x_2) = \sigma^2 \exp\left(-\frac{1}{2l^2}(x_1 - x_2)^2\right)$$



Hyperparameters significantly influence outcome

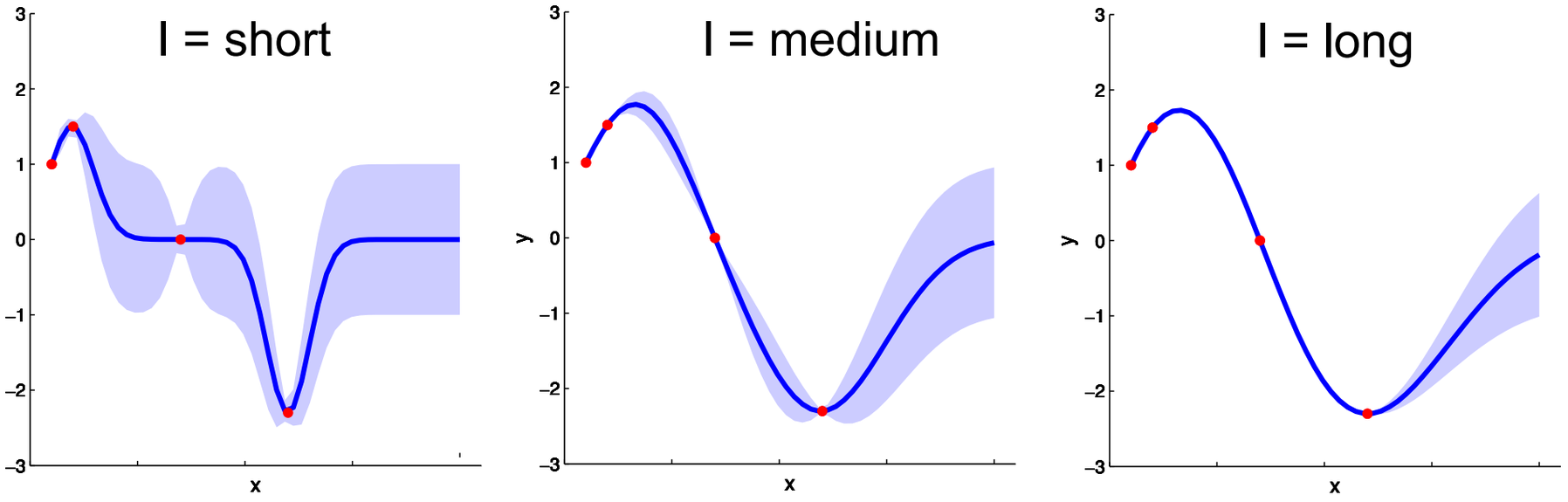
Learn values from data!

A. Maximum likelihood fit:

$$\text{argmax}_\theta \text{P}(\mathbf{y}|\theta)$$

Quick recap
4. GP hyperparameters

$$K(x_1, x_2) = \sigma^2 \exp\left(-\frac{1}{2l^2}(x_1 - x_2)^2\right)$$



Hyperparameters significantly influence outcome

Learn values from data!

A. Maximum likelihood fit:

$$\text{argmax}_\theta \text{P}(\mathbf{y}|\theta)$$

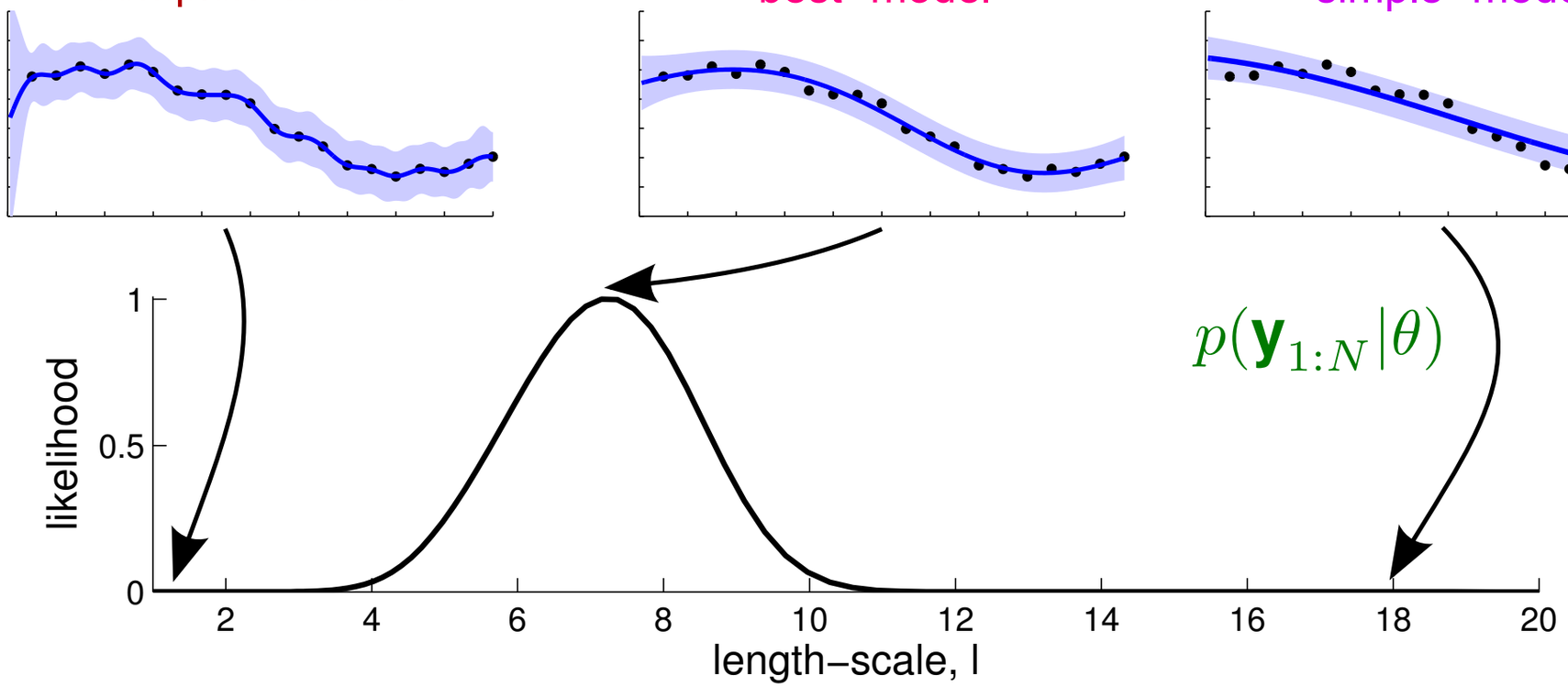
B. Bayesian:

$$p(\theta|\mathbf{y}_{1:N}) = \frac{p(\mathbf{y}_{1:N}|\theta)p(\theta)}{p(\mathbf{y}_{1:N})}$$

fits every training point
"complex" model

"best" model

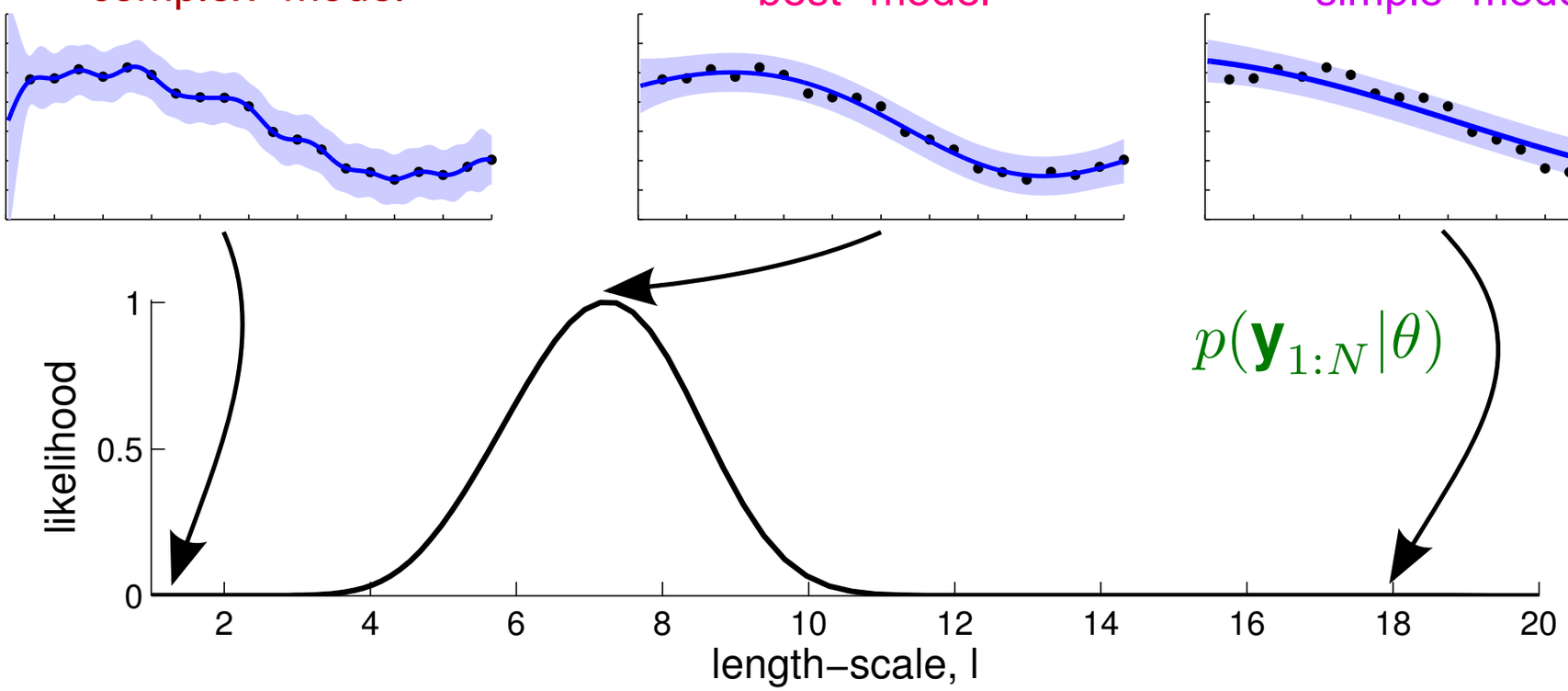
straight line-like
"simple" model

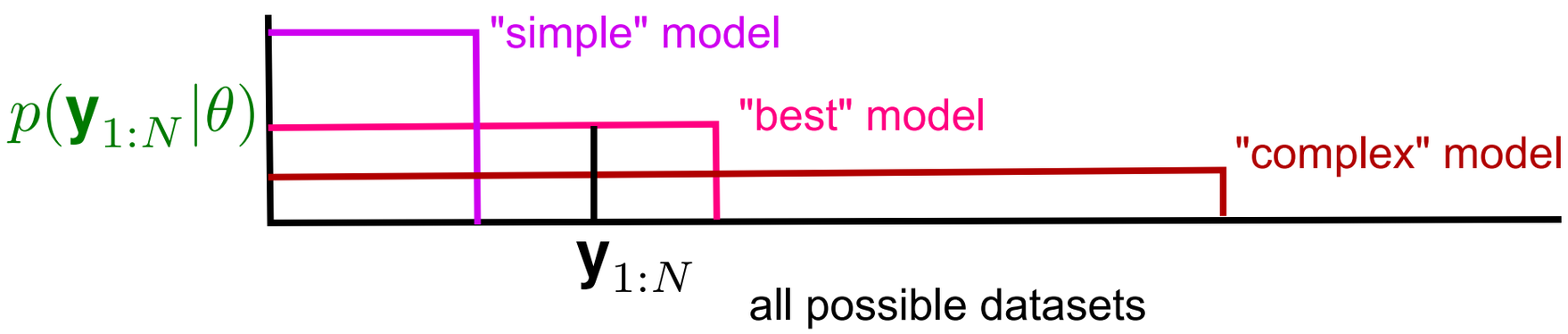


fits every training point
"complex" model

"best" model

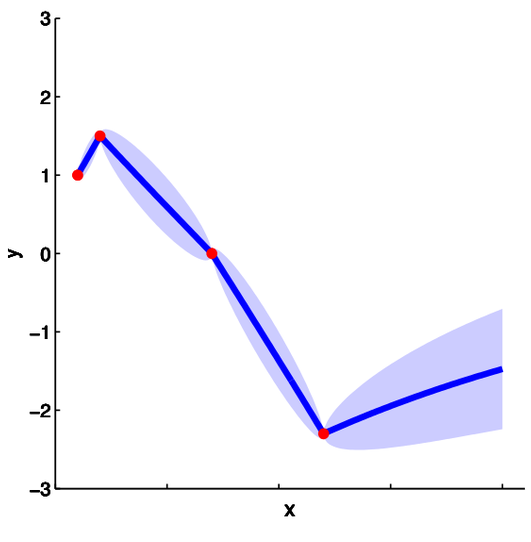
straight line-like
"simple" model



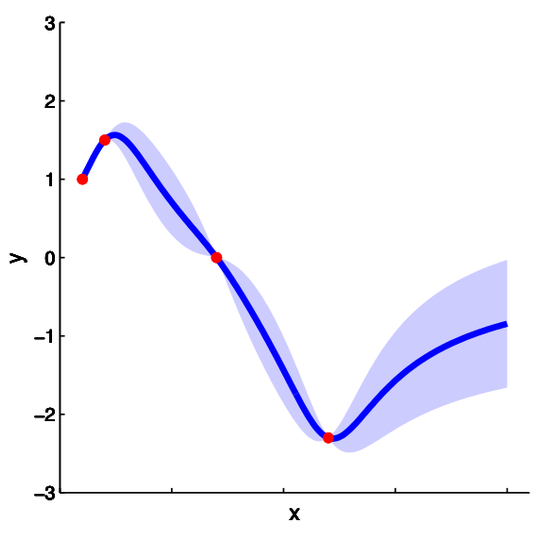


Quick recap
5. GP kernels

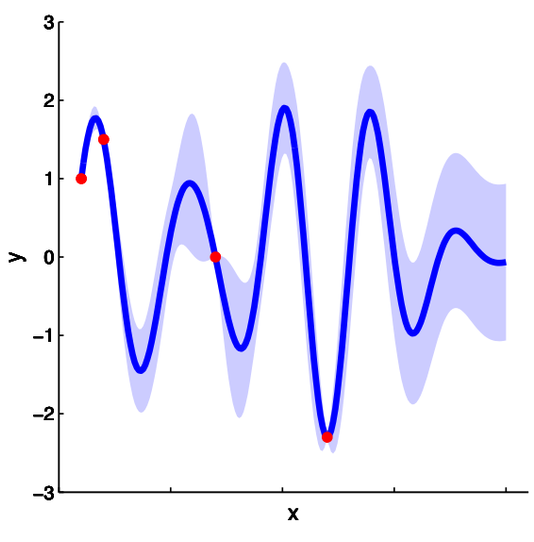
OU



RQ

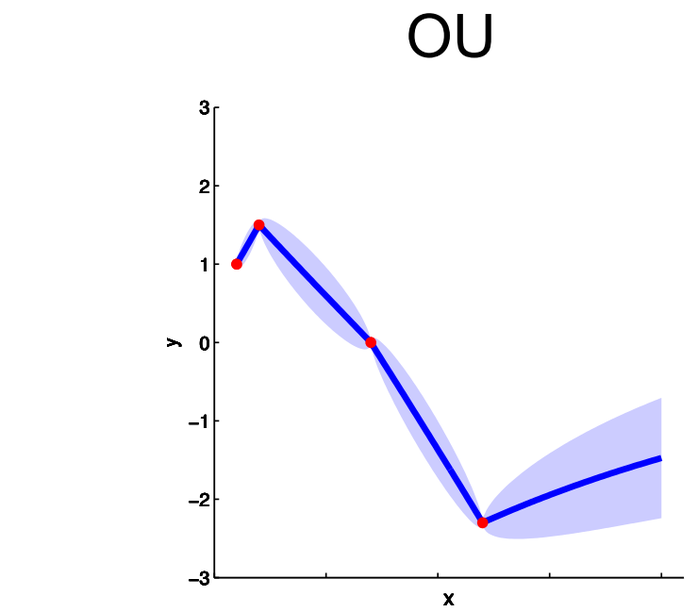


periodic



Choice of covariance function influence outcome

Quick recap
5. GP kernels

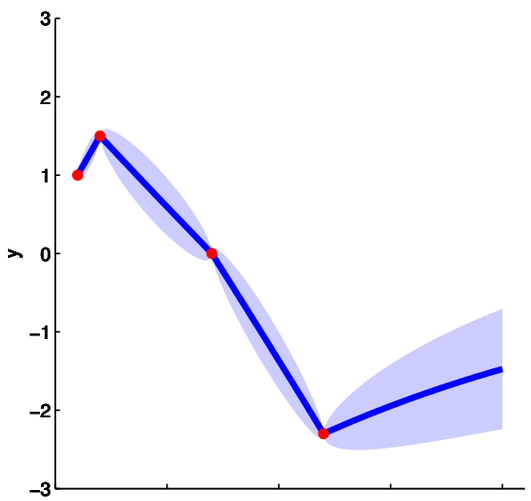


Choice of covariance function influence outcome

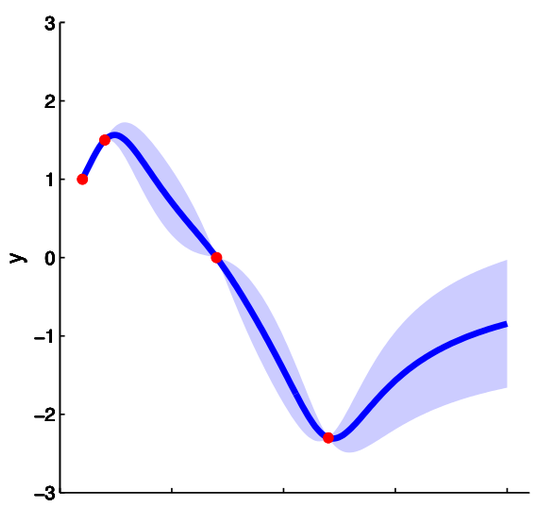
$$p(M|\mathbf{y}_{1:N}) = \frac{p(\mathbf{y}_{1:N}|M)p(M)}{\sum_{M'} p(\mathbf{y}_{1:N}|M')p(M')}$$

Quick recap
5. GP kernels

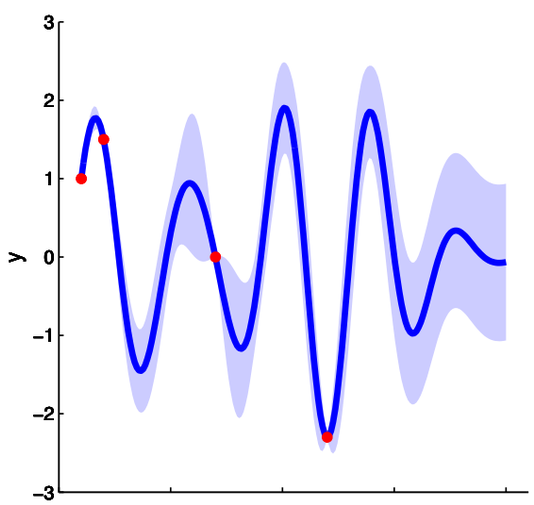
OU



RQ



periodic

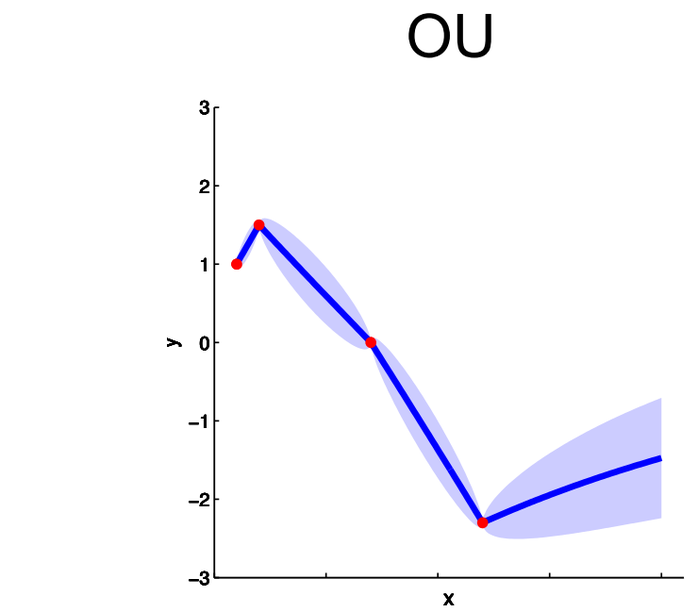


Choice of covariance function influence outcome

Prior over models

$$p(M|\mathbf{y}_{1:N}) = \frac{p(\mathbf{y}_{1:N}|M)p(M)}{\sum_{M'} p(\mathbf{y}_{1:N}|M')p(M')}$$

Quick recap
5. GP kernels

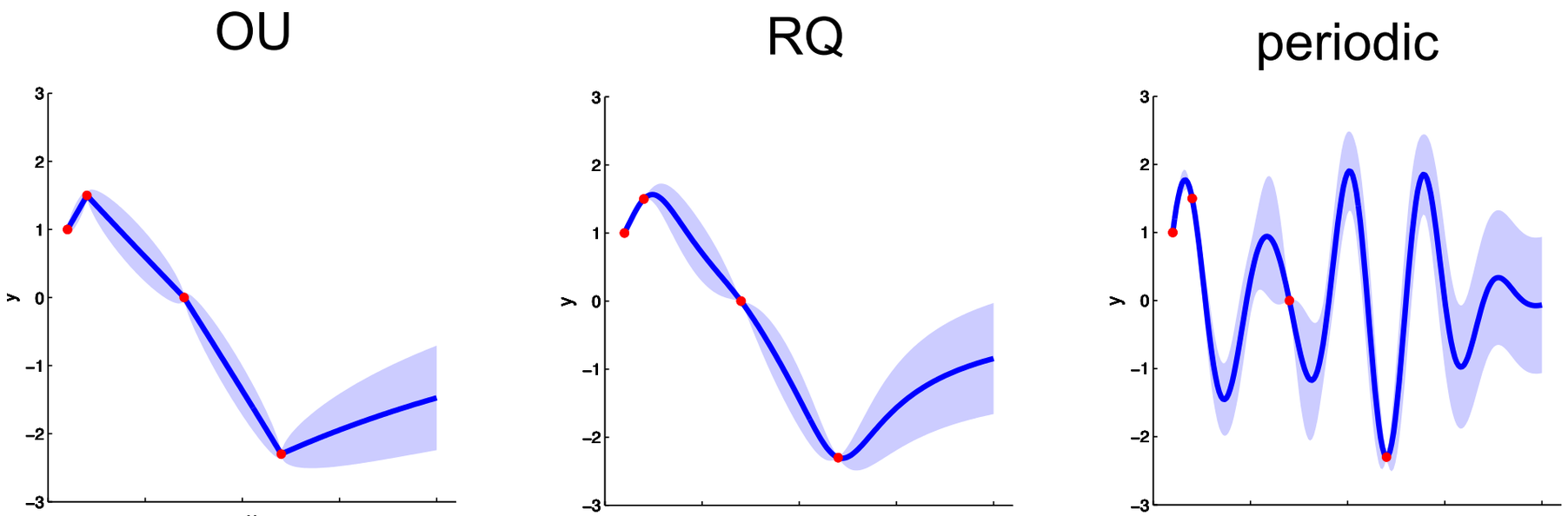


Choice of covariance function influence outcome

Marginal likelihood Prior over models

$$p(M|\mathbf{y}_{1:N}) = \frac{p(\mathbf{y}_{1:N}|M)p(M)}{\sum_{M'} p(\mathbf{y}_{1:N}|M')p(M')}$$

Quick recap
5. GP kernels



Choice of covariance function influence outcome

Marginal likelihood Prior over models

$$p(M|\mathbf{y}_{1:N}) = \frac{p(\mathbf{y}_{1:N}|M)p(M)}{\sum_{M'} p(\mathbf{y}_{1:N}|M')p(M')}$$

$$p(\mathbf{y}_{1:N}|M) = \int \mathrm{d}\theta \; p(\mathbf{y}_{1:N}|\theta, M)p(\theta|M)$$

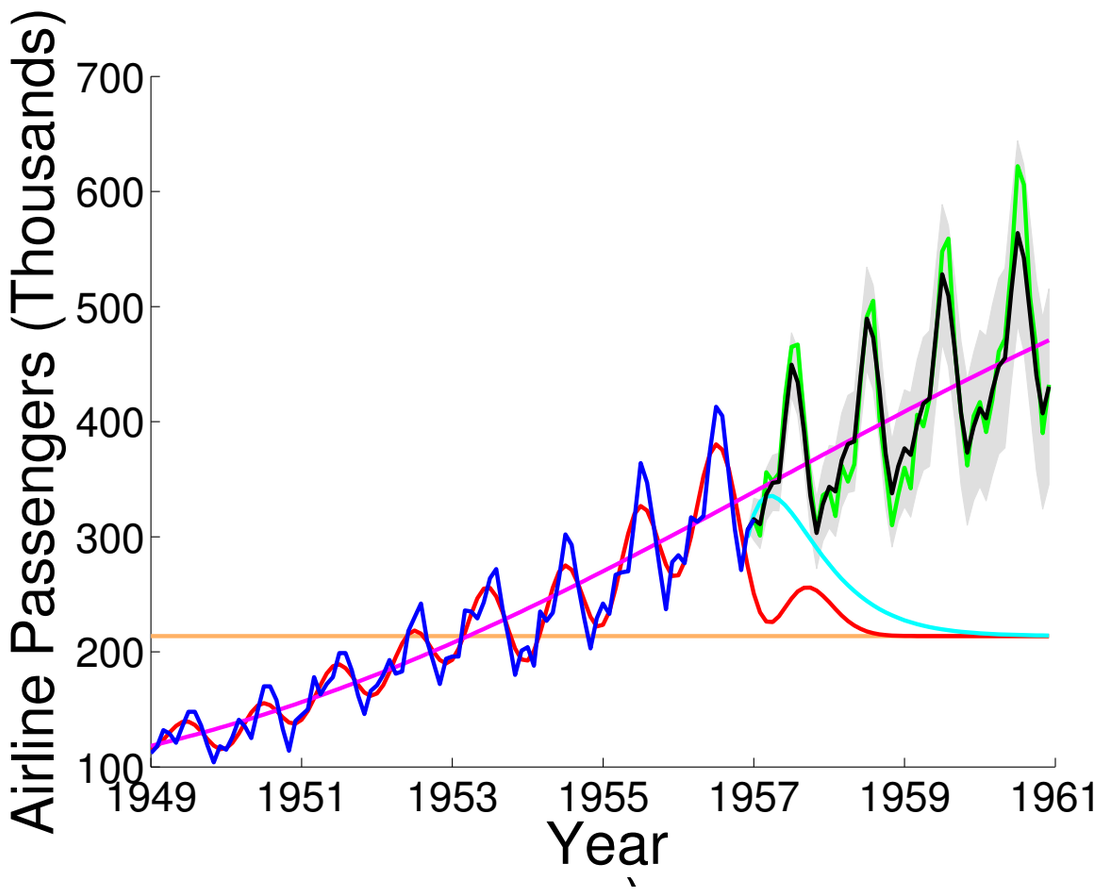
Usually unpleasant

Meta-kernel, learn which is right directly

$$k(\tau) = \sum_{q=1}^{Q} w_q \prod_{p=1}^{P} \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\} \cos(2\pi \tau_p \mu_q^{(p)}).$$

Meta-kernel, learn which is right directly

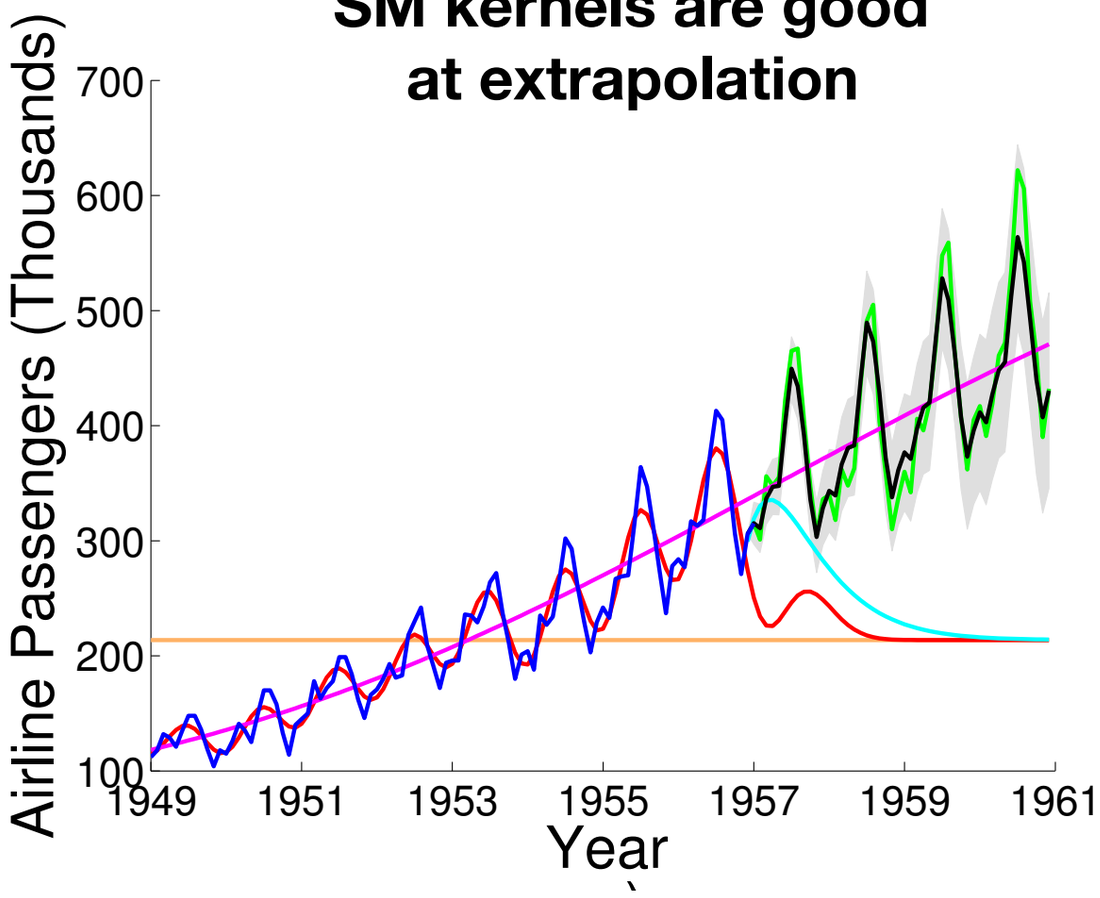
$$k(\tau) = \sum_{q=1}^{Q} w_q \prod_{p=1}^{P} \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\} \cos(2\pi \tau_p \mu_q^{(p)}).$$



Meta-kernel, learn which is right directly

$$k(\tau) = \sum_{q=1}^{Q} w_q \prod_{p=1}^{P} \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\} \cos(2\pi \tau_p \mu_q^{(p)}).$$

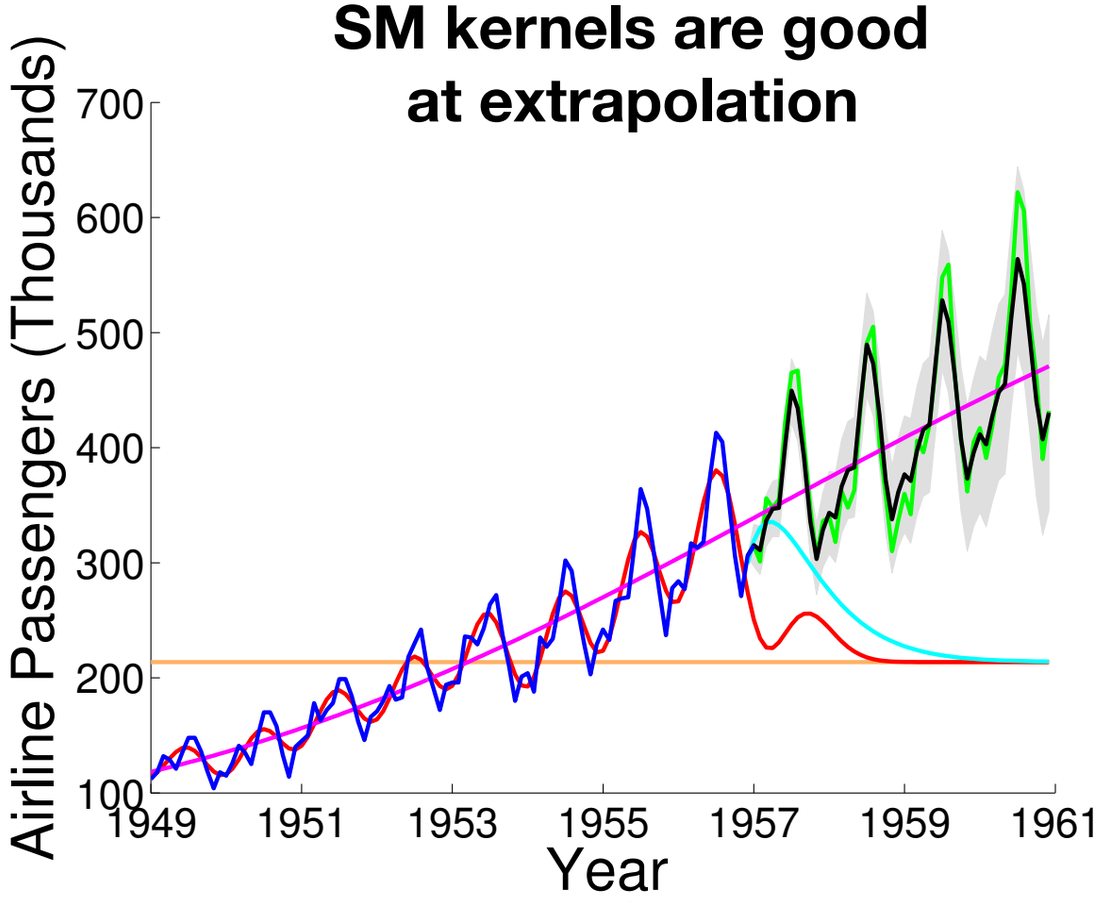
SM kernels are good at extrapolation

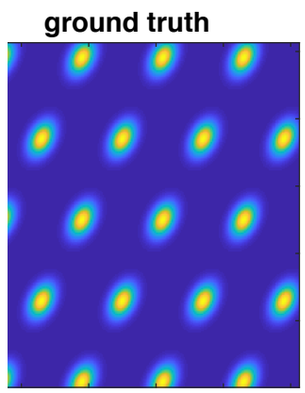


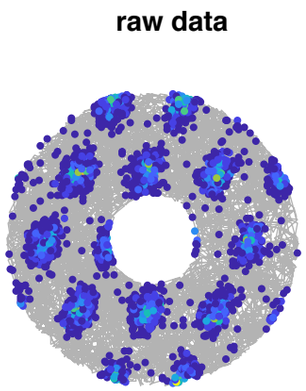
Meta-kernel, learn which is right directly

$$k(\tau) = \sum_{q=1}^{Q} w_q \prod_{p=1}^{P} \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\} \cos(2\pi \tau_p \mu_q^{(p)}).$$

SM kernels are good at extrapolation



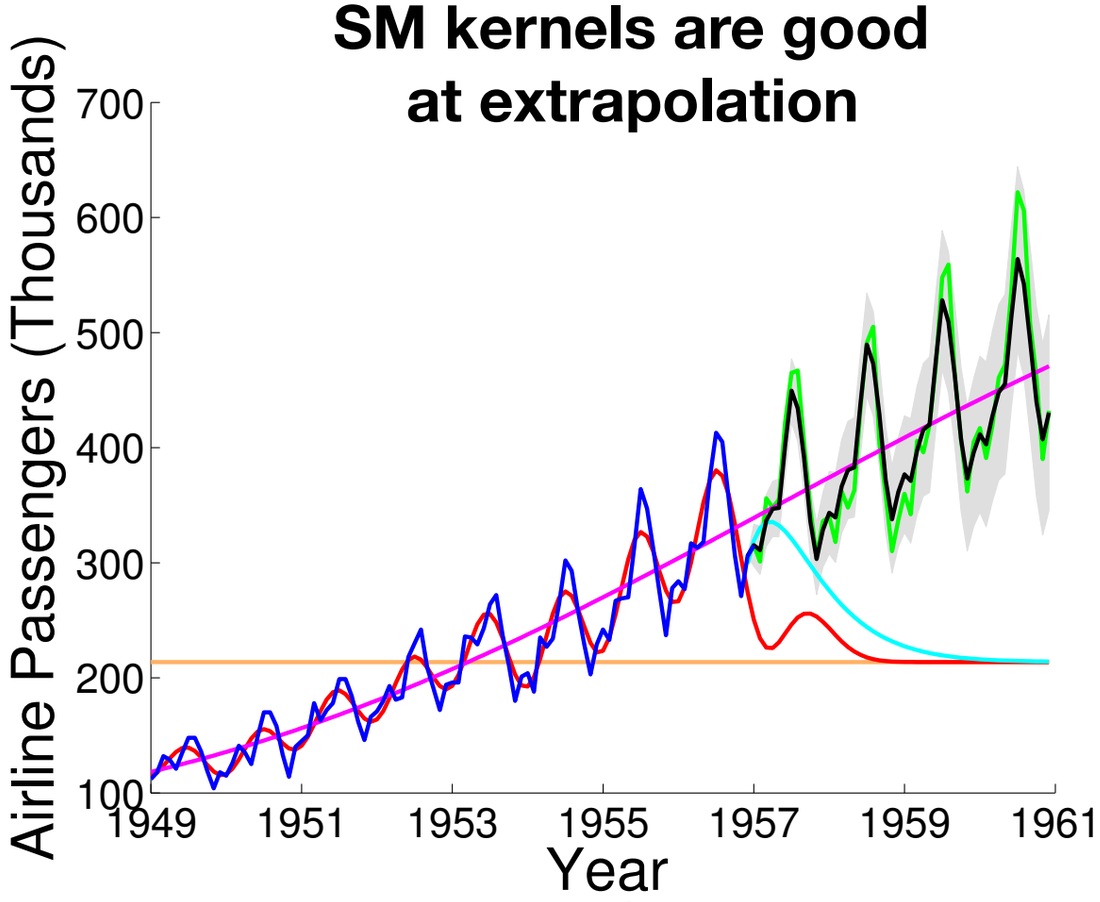


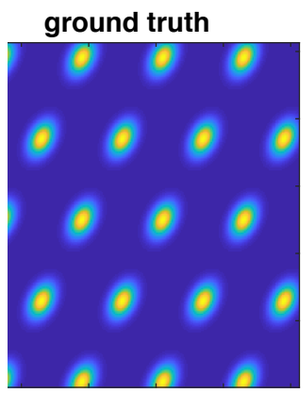


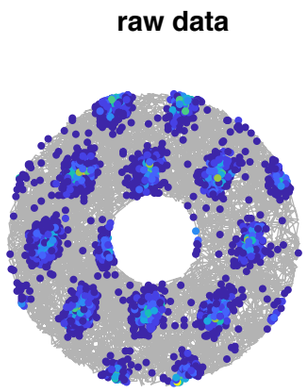
Meta-kernel, learn which is right directly

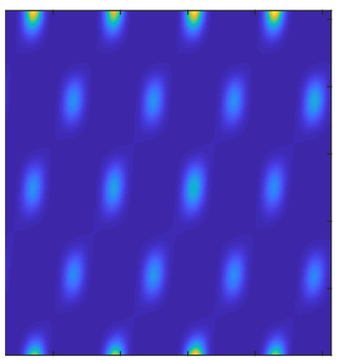
$$k(\tau) = \sum_{q=1}^{Q} w_q \prod_{p=1}^{P} \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\} \cos(2\pi \tau_p \mu_q^{(p)}).$$

SM kernels are good at extrapolation





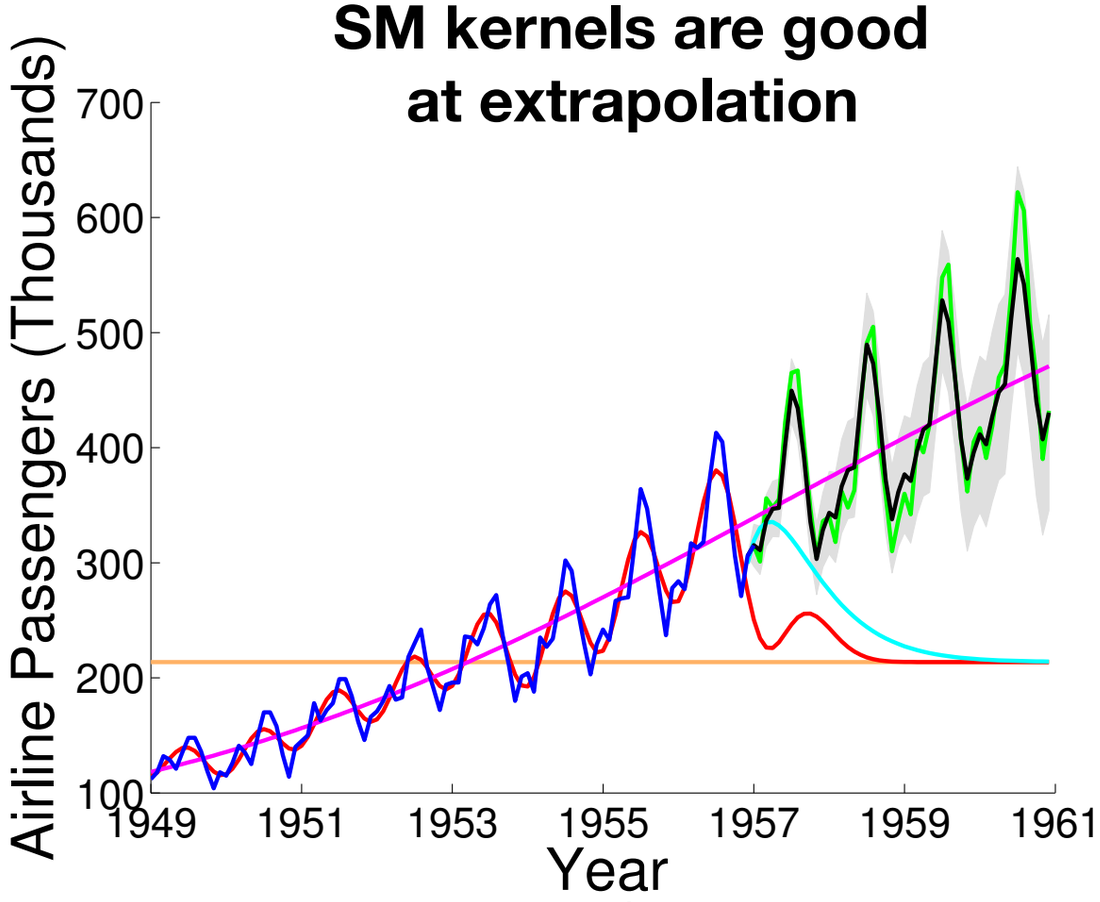


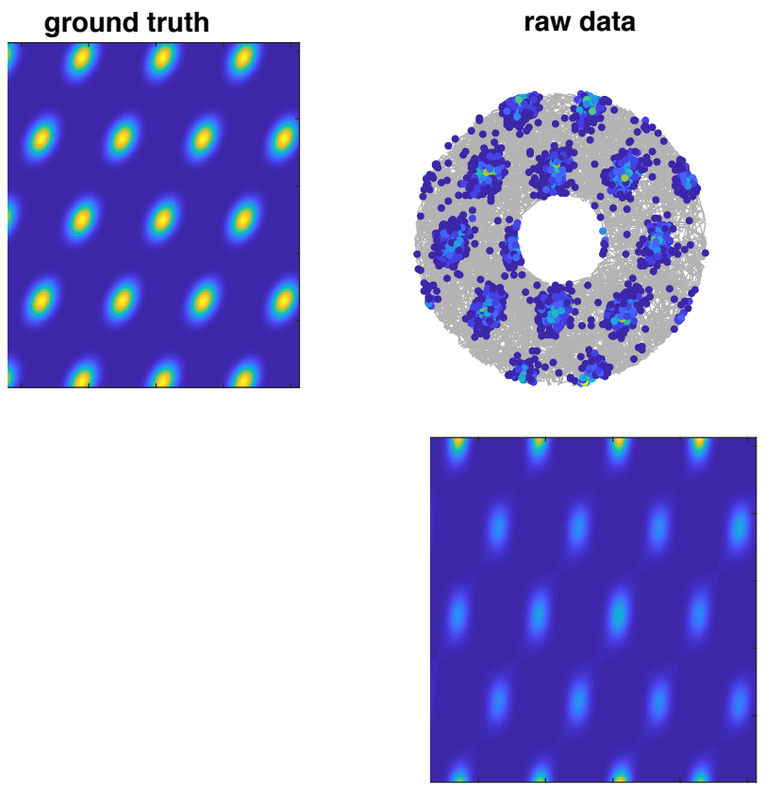


Meta-kernel, learn which is right directly

$$k(\tau) = \sum_{q=1}^{Q} w_q \prod_{p=1}^{P} \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\} \cos(2\pi \tau_p \mu_q^{(p)}).$$

SM kernels are good at extrapolation



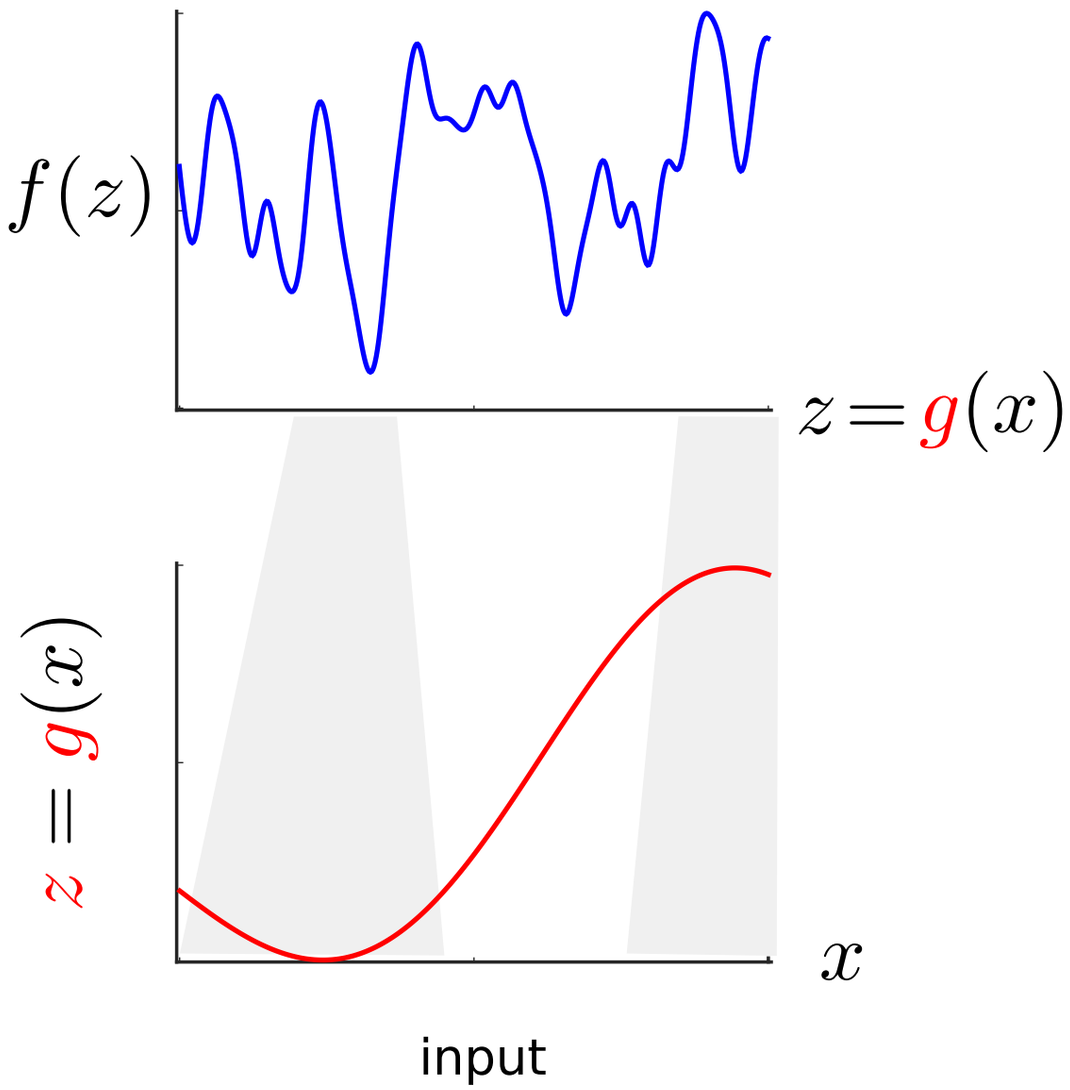


Wilson, Flaxman, Savin

Deep GPs

$$y(x) = f(g(x)) + \sigma_{\mathrm{y}}\epsilon$$

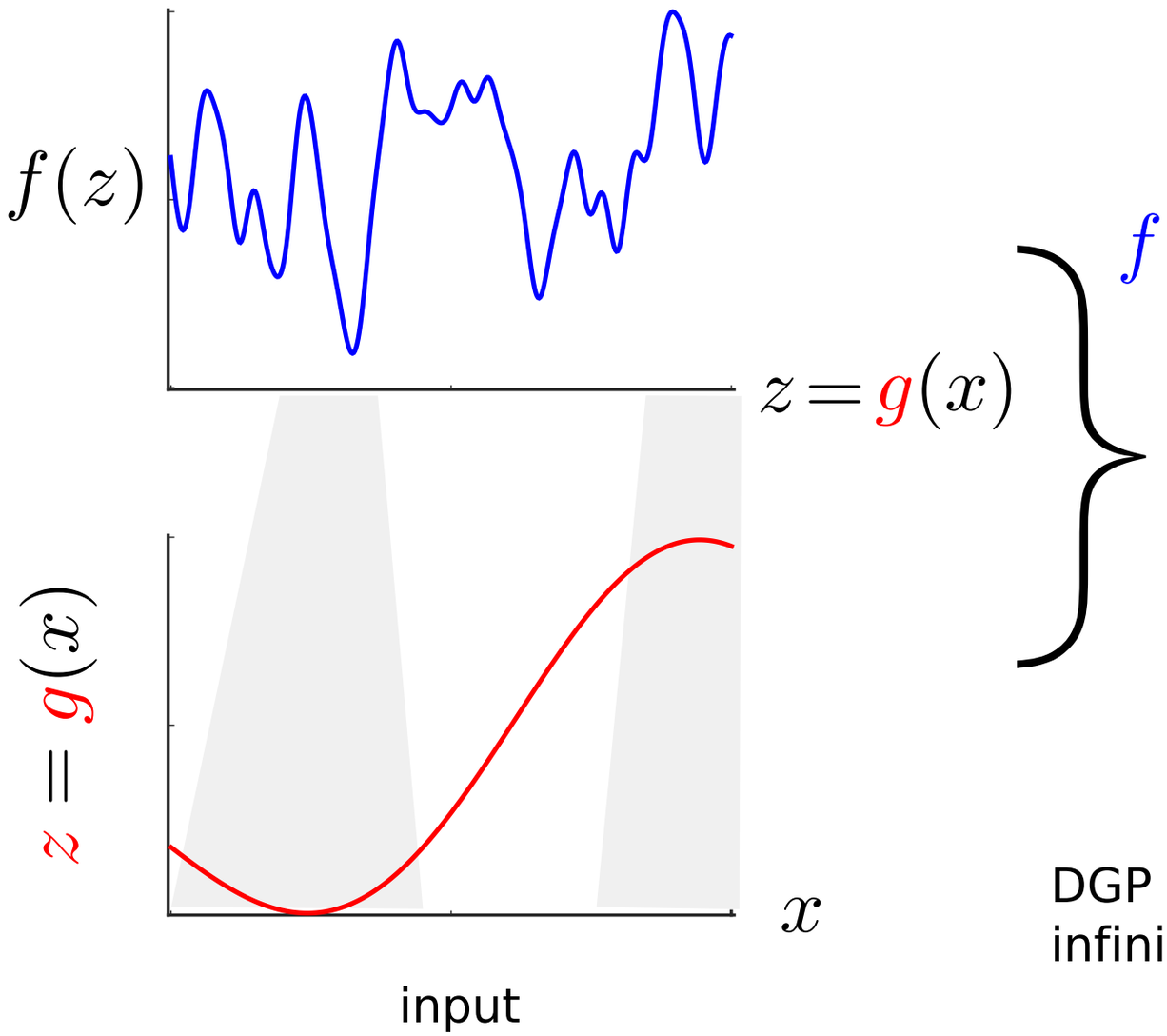
$$f(x) = \mathcal{GP}(0, K_f(x, x'))$$
$$g(x) = \mathcal{GP}(0, K_g(x, x'))$$

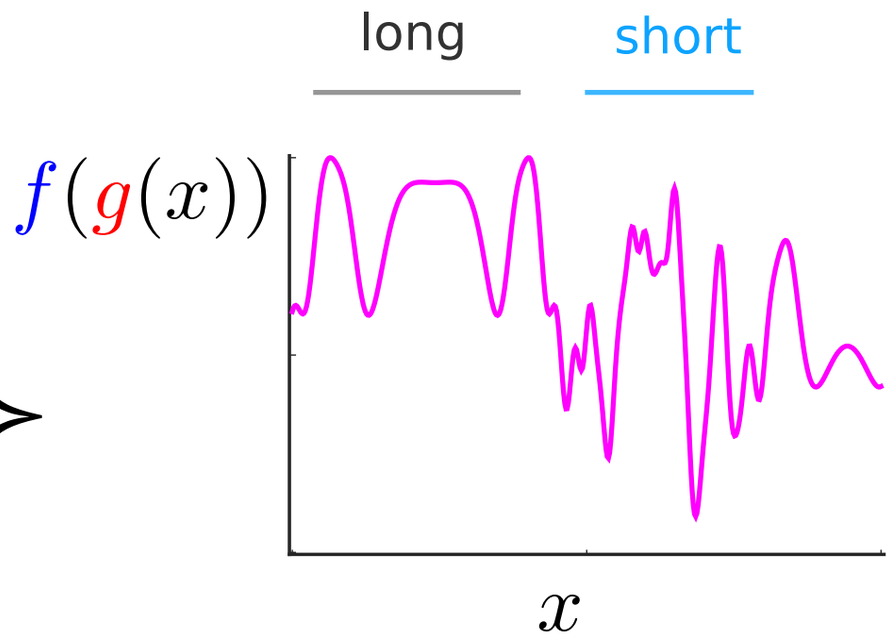


Deep GPs

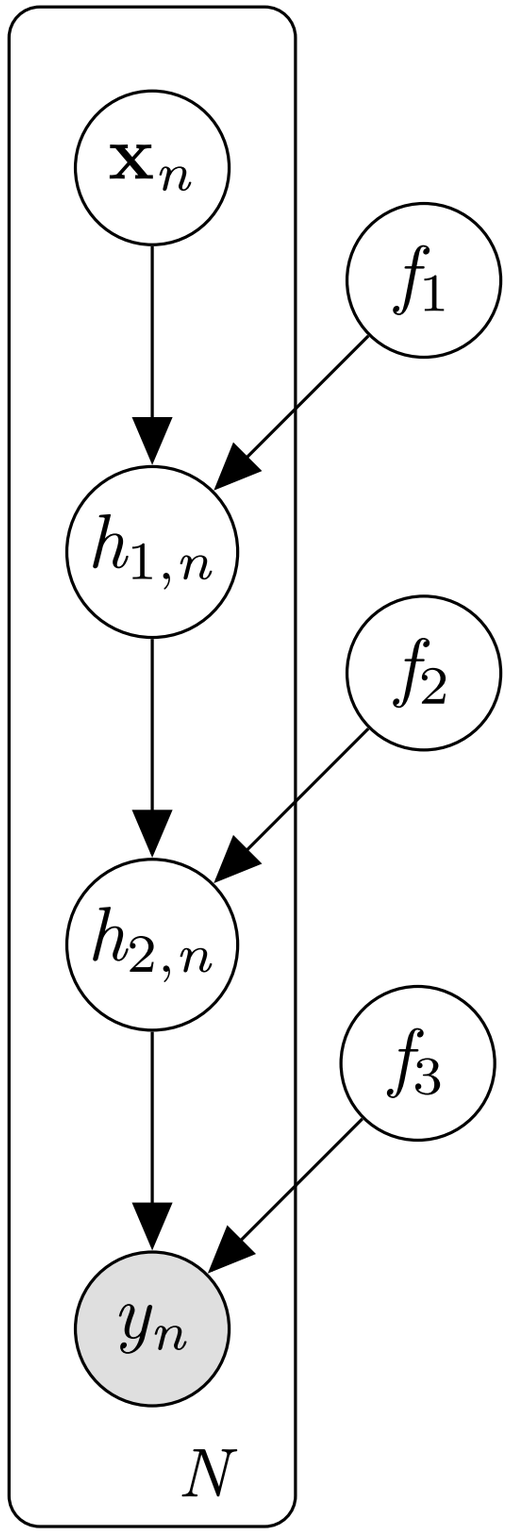
$$y(x) = f(g(x)) + \sigma_{\mathrm{y}}\epsilon$$

$$f(x) = \mathcal{GP}(0, K_f(x, x'))$$
$$g(x) = \mathcal{GP}(0, K_g(x, x'))$$

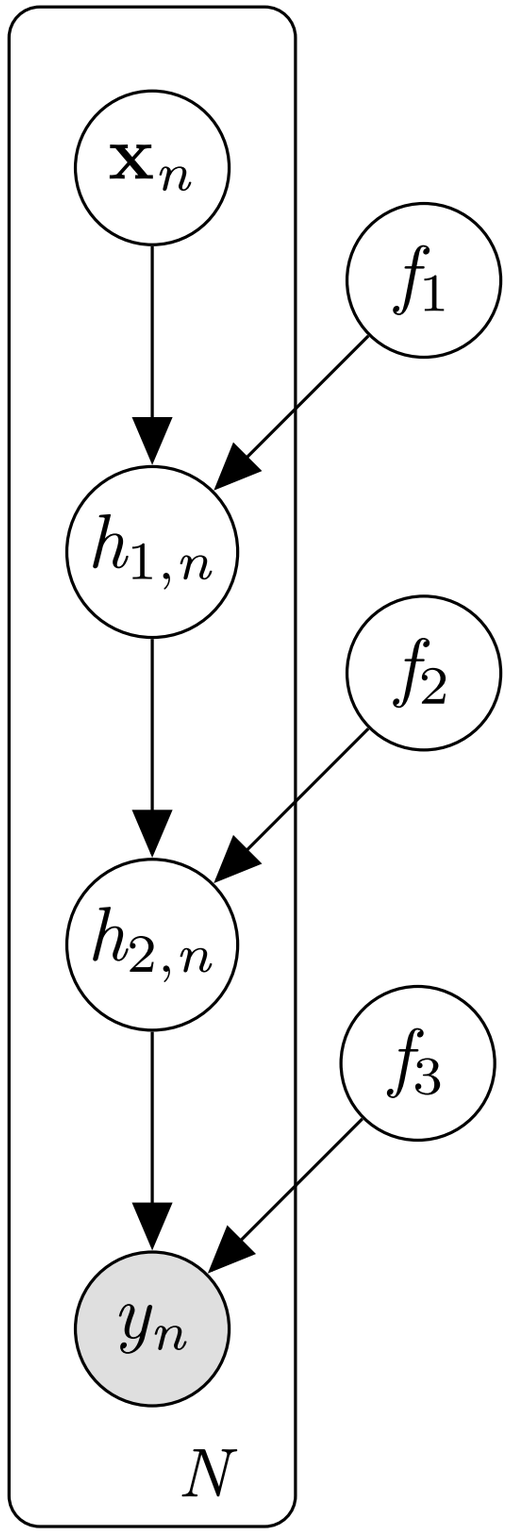




DGP = multi-layer neural network,
infinitely wide hidden layer



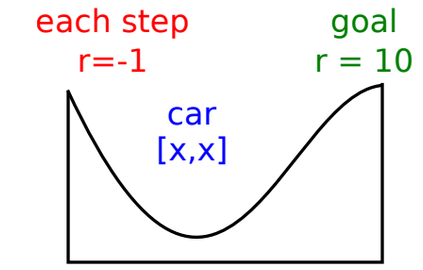
$$f_l \sim \mathcal{GP}(0, k(.,.))$$
$$y_n = g(\mathbf{x}_n) = f_L(f_{L-1}(\cdots f_2(f_1(\mathbf{x}_n)))) + \epsilon_n$$
$$h_{L-1,n} := f_{L-1}(\cdots f_1(\mathbf{x}_n)), y_n = f_L(h_{L-1,n}) + \epsilon_n$$



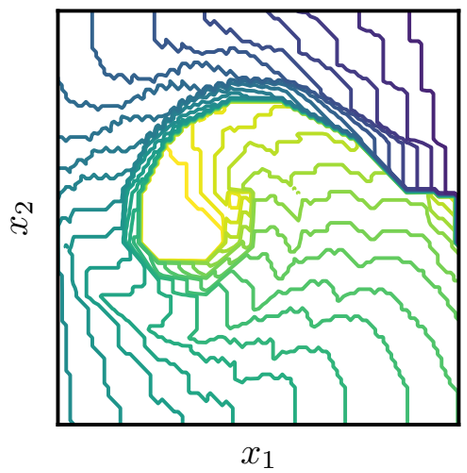
$$f_l \sim \mathcal{GP}(0, k(.,.))$$

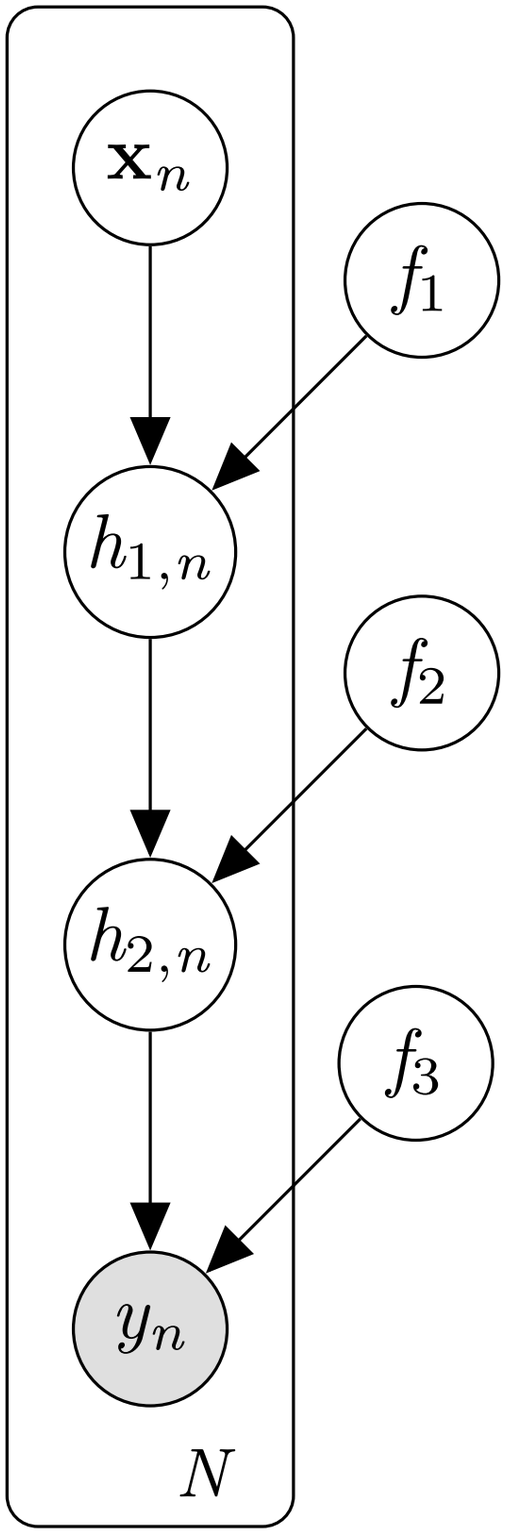
$$y_n = g(\mathbf{x}_n) = f_L(f_{L-1}(\cdots f_2(f_1(\mathbf{x}_n)))) + \epsilon_n$$

$$h_{L-1,n} := f_{L-1}(\cdots f_1(\mathbf{x}_n)), y_n = f_L(h_{L-1,n}) + \epsilon_n$$



Value function

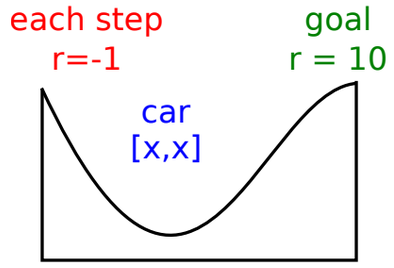




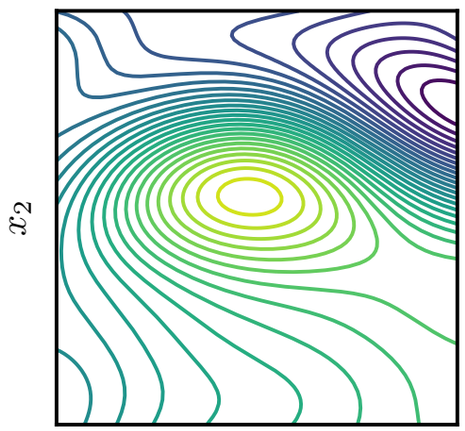
$$f_l \sim \mathcal{GP}(0, k(.,.))$$

$$y_n = g(\mathbf{x}_n) = f_L(f_{L-1}(\cdots f_2(f_1(\mathbf{x}_n)))) + \epsilon_n$$

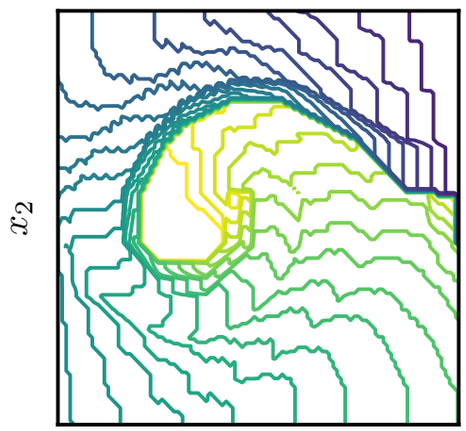
$$h_{L-1,n} := f_{L-1}(\cdots f_1(\mathbf{x}_n)), y_n = f_L(h_{L-1,n}) + \epsilon_n$$

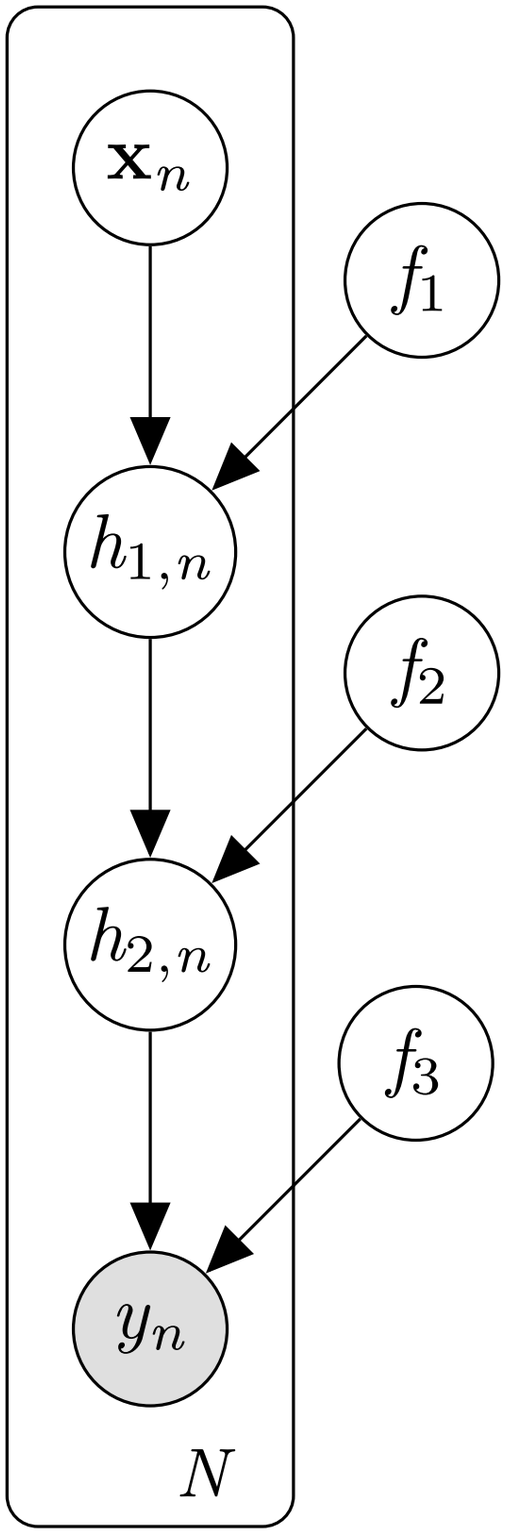


GP fit



Value function

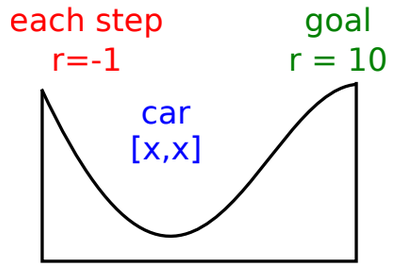


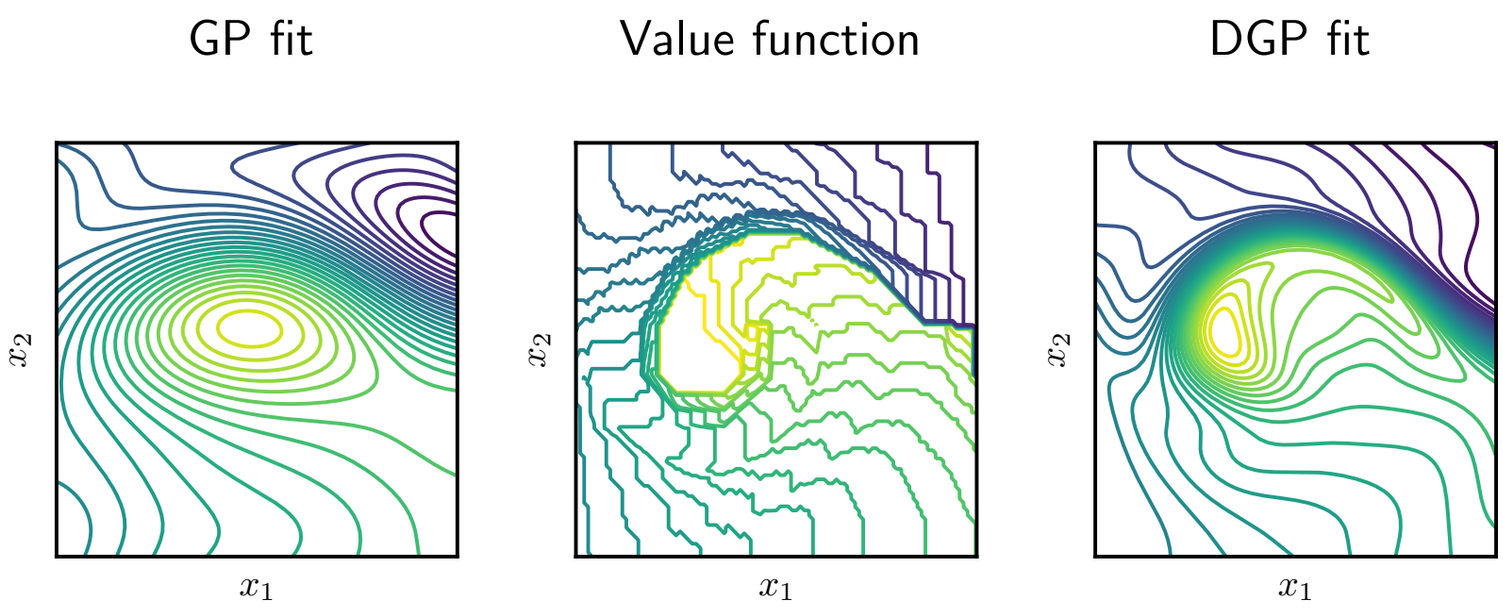


$$f_l \sim \mathcal{GP}(0, k(.,.))$$

$$y_n = g(\mathbf{x}_n) = f_L(f_{L-1}(\cdots f_2(f_1(\mathbf{x}_n)))) + \epsilon_n$$

$$h_{L-1,n} := f_{L-1}(\cdots f_1(\mathbf{x}_n)), y_n = f_L(h_{L-1,n}) + \epsilon_n$$





What are Gaussian Processes good for?

What are Gaussian Processes good for?

Strengths

What are Gaussian Processes good for?

Strengths

interpretable: covariance functions specify easy-to-explain high-level properties of functions

What are Gaussian Processes good for?

Strengths

interpretable: covariance functions specify easy-to-explain high-level properties of functions

data-efficient: non-parametric + Bayesian ⟹ lots of flexibility + avoid overfitting

What are Gaussian Processes good for?

Strengths

interpretable: covariance functions specify easy-to-explain high-level properties of functions

data-efficient: non-parametric + Bayesian ⟹ lots of flexibility + avoid overfitting

optimal decision making: well-calibrated uncertainties: knows when it does not know

What are Gaussian Processes good for?

Strengths

interpretable: covariance functions specify easy-to-explain high-level properties of functions

data-efficient: non-parametric + Bayesian ⟹ lots of flexibility + avoid overfitting

optimal decision making: well-calibrated uncertainties: knows when it does not know

Weaknesses

What are Gaussian Processes good for?

Strengths

interpretable: covariance functions specify easy-to-explain high-level properties of functions

data-efficient: non-parametric + Bayesian ⟹ lots of flexibility + avoid overfitting

optimal decision making: well-calibrated uncertainties: knows when it does not know

Weaknesses

large datasets: $N \leq 10^5$ unless there is special structure (invert and store covariance matrix)

What are Gaussian Processes good for?

Strengths

interpretable: covariance functions specify easy-to-explain high-level properties of functions

data-efficient: non-parametric + Bayesian ⟹ lots of flexibility + avoid overfitting

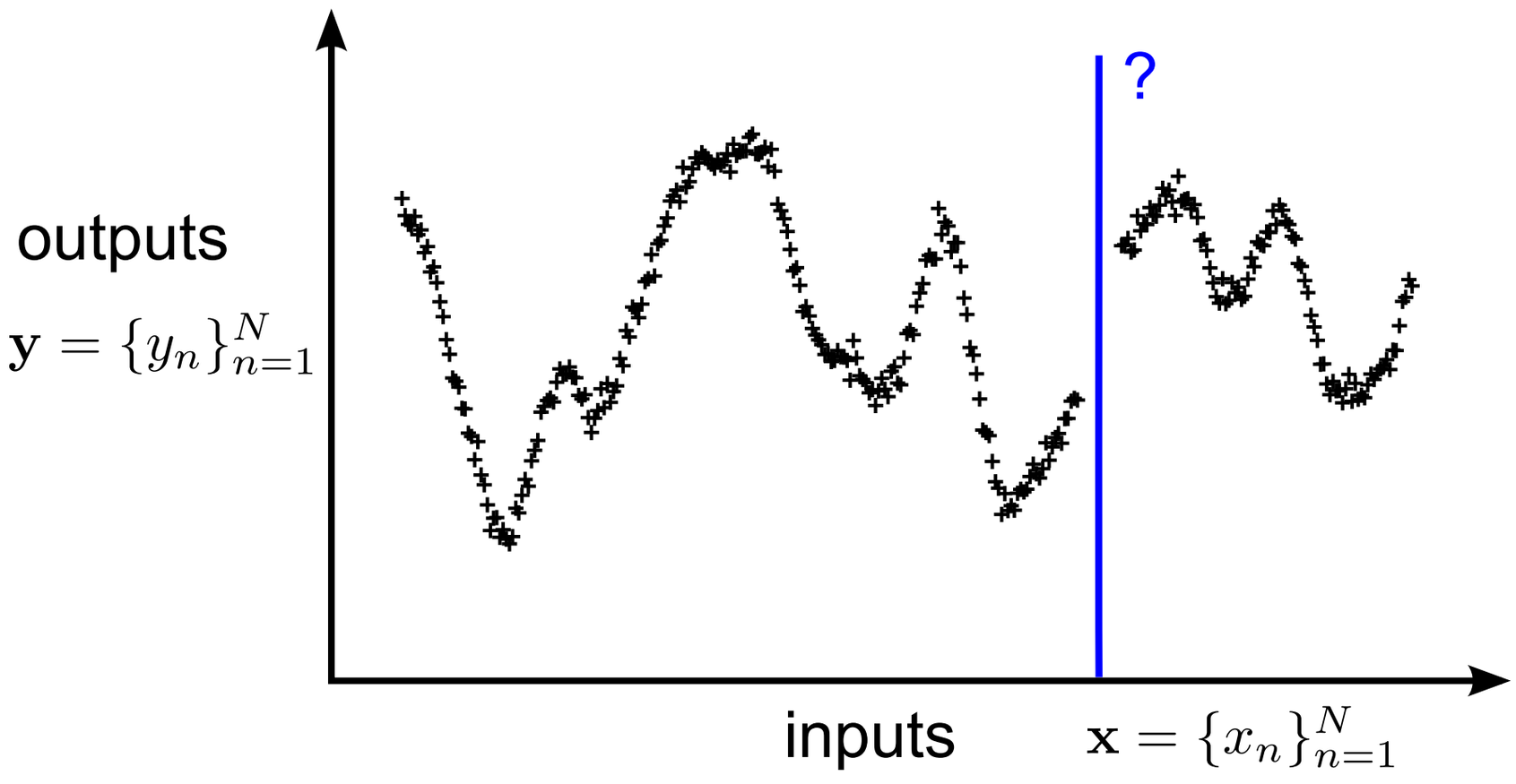
optimal decision making: well-calibrated uncertainties: knows when it does not know

Weaknesses

large datasets: $N \leq 10^5$ unless there is special structure (invert and store covariance matrix)

high-dimensional inputs spaces: $D \leq 10^2$ unless there is special structure, e.g. Kronecker (compute pair-wise elements of covariance function)

How do me make these work in practice?



How do me make these work in practice?

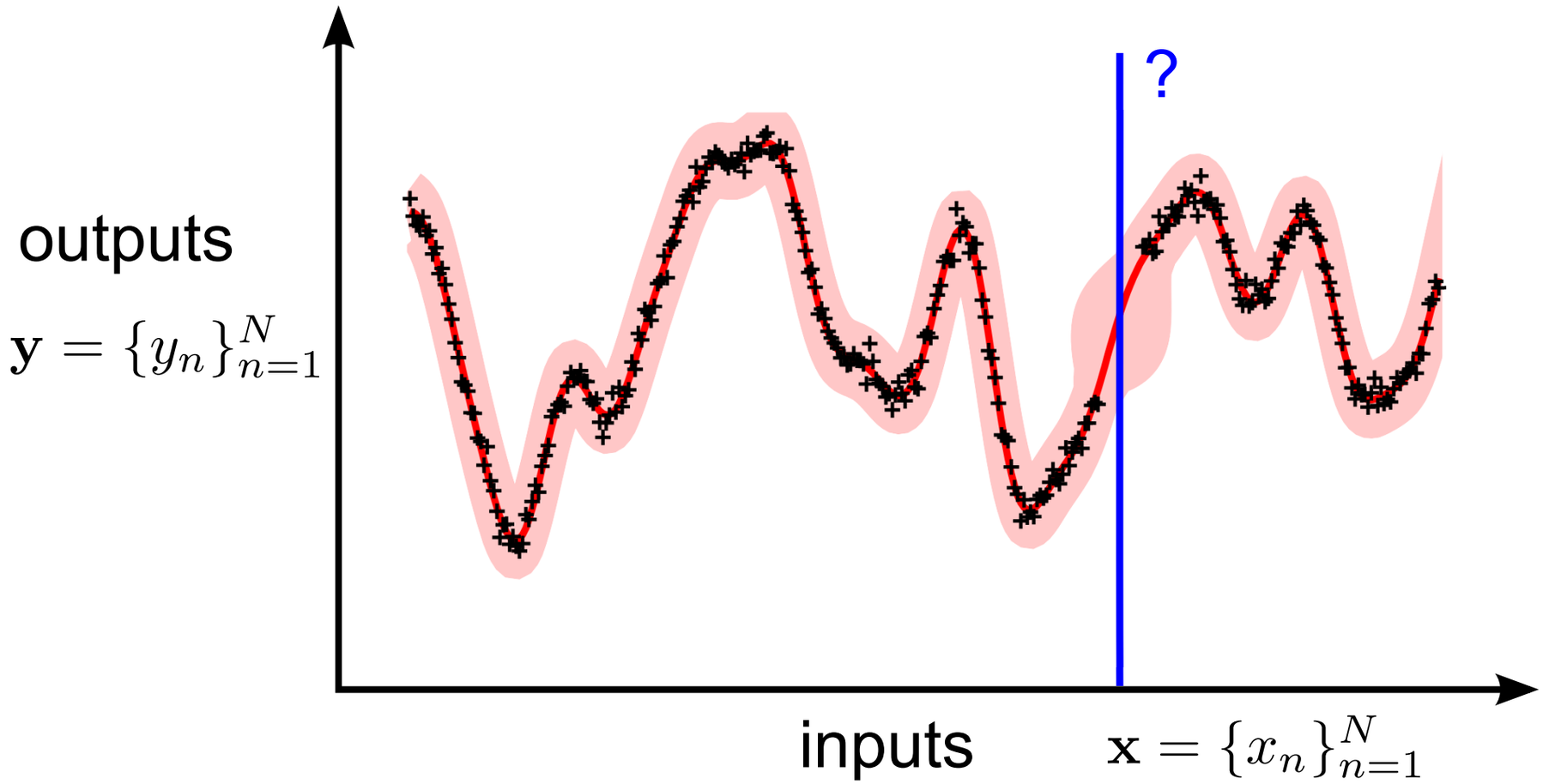
$$p(f|\theta) = \mathcal{GP}(f; 0, \mathrm{K}_\theta)$$

$$p(y_n|f, x_n, \theta)$$

inference & learning

$$p(f|\mathbf{y}, \mathbf{x}, \theta)$$

$$p(\mathbf{y}|\mathbf{x}, \theta)$$



How do me make these work in practice?

$$p(f|\theta) = \mathcal{GP}(f; 0, \mathrm{K}_\theta)$$

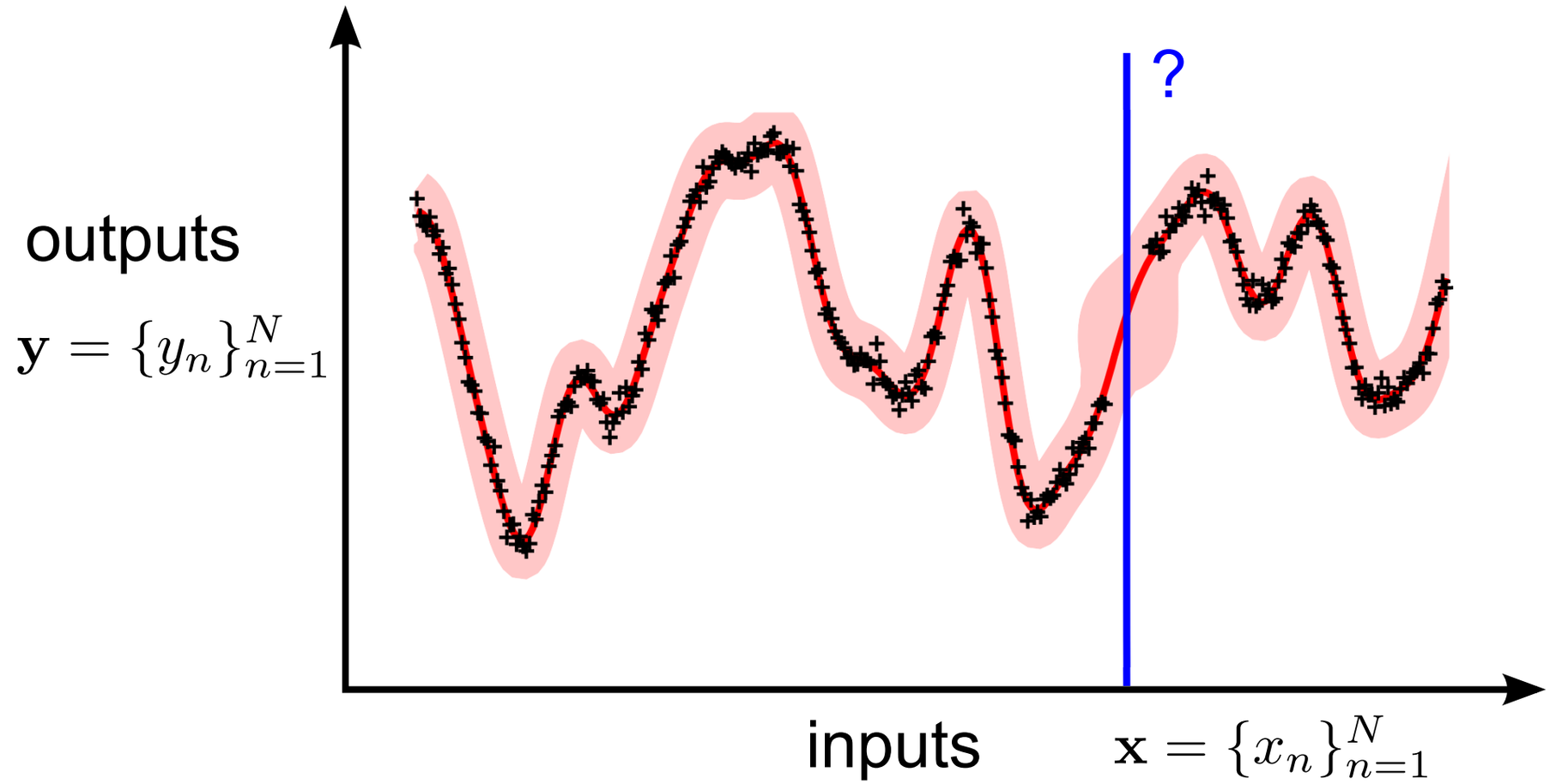
$$p(y_n|f, x_n, \theta)$$

inference & learning

intractabilities
computational $\mathcal{O}(N^3)$
analytic

$$p(f|\mathbf{y}, \mathbf{x}, \theta)$$

$$p(\mathbf{y}|\mathbf{x}, \theta)$$



$$p(f|\theta) = \mathcal{GP}(f; 0, \mathrm{K}_\theta)$$

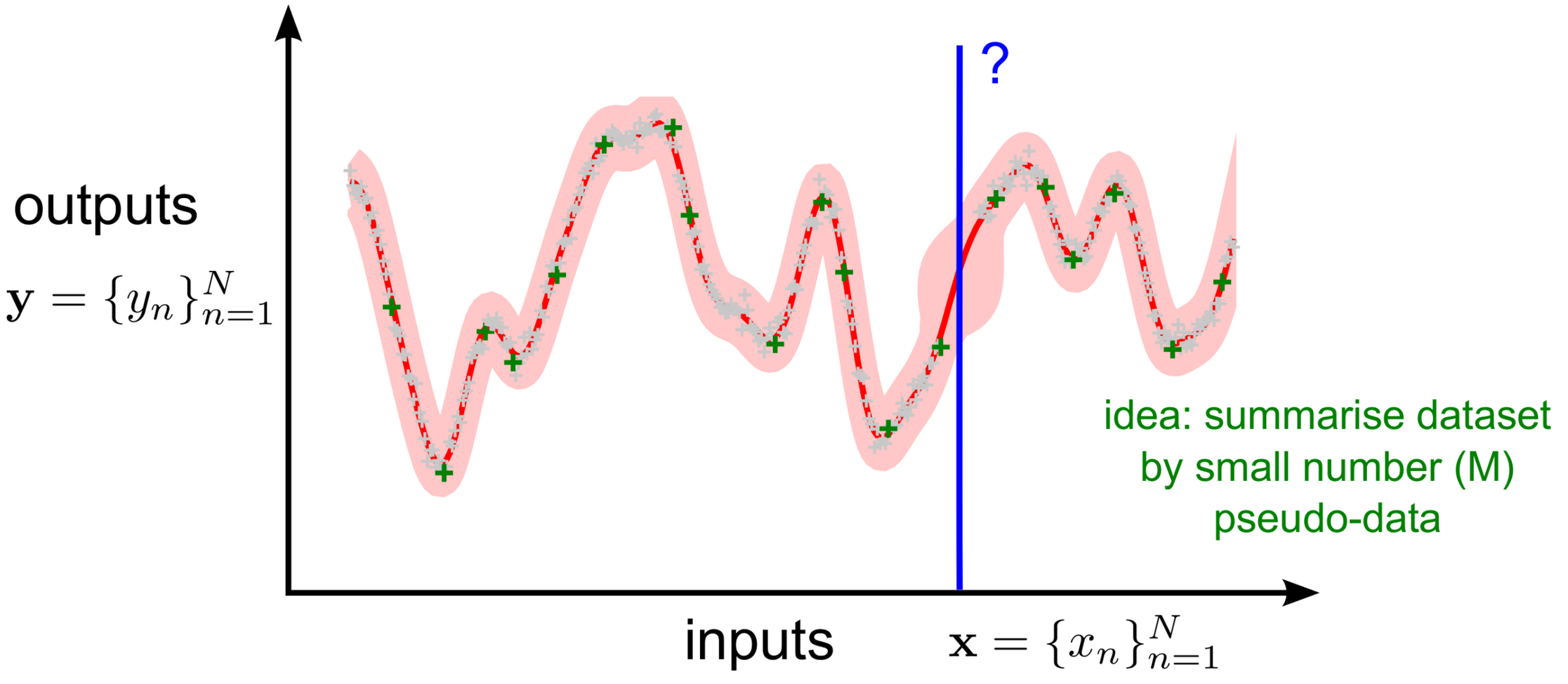
$$p(y_n|f, x_n, \theta)$$

inference & learning

intractabilities
computational $\mathcal{O}(N^3)$
analytic

$$p(f|\mathbf{y}, \mathbf{x}, \theta)$$

$$p(\mathbf{y}|\mathbf{x}, \theta)$$



A Brief History of Gaussian Process Approximations

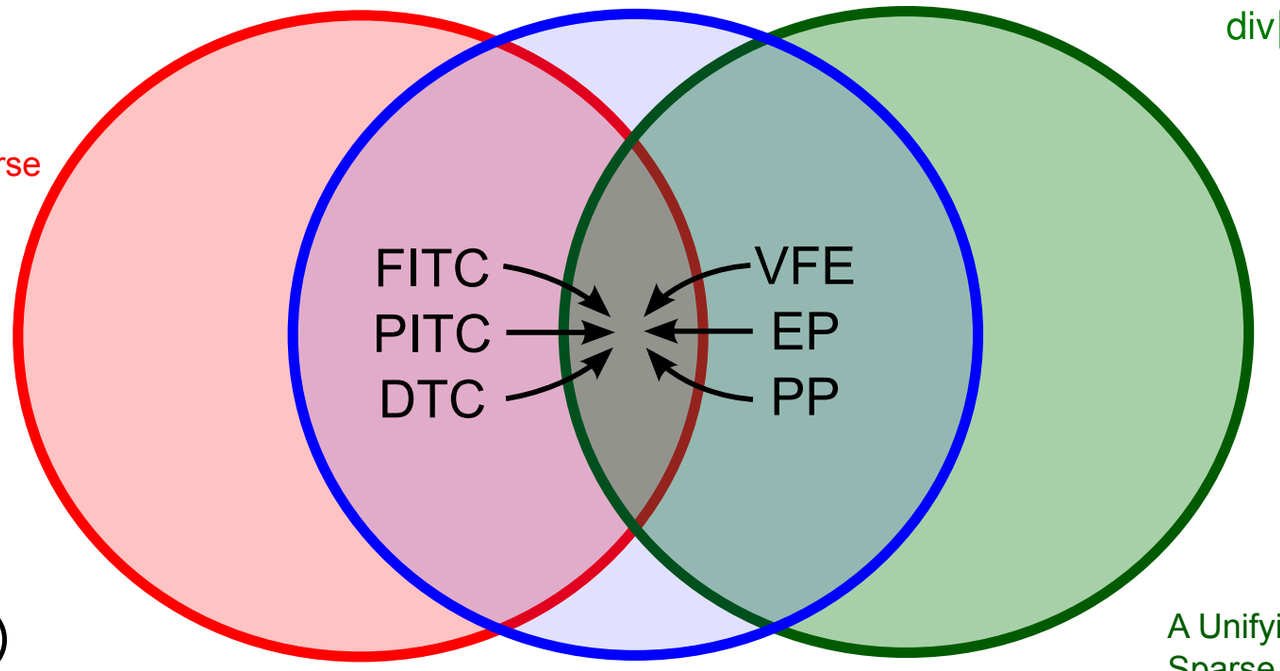
approximate generative model
exact inference

methods employing
pseudo-data

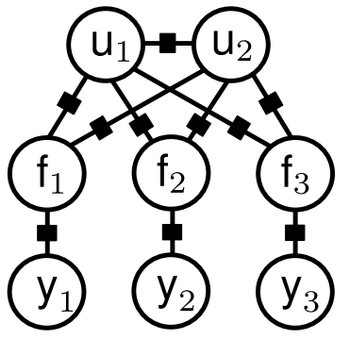
exact generative model
approximate inference

$\text{div}[p(\mathbf{f},\mathbf{y})||q(\mathbf{f},\mathbf{y})]$

$\text{div}[p(\mathbf{f}|\mathbf{y})||q(\mathbf{f})]$



A Unifying View of Sparse Approximate Gaussian Process Regression
Quinonero-Candela & Rasmussen, 2005
(FITC, PITC, DTC)



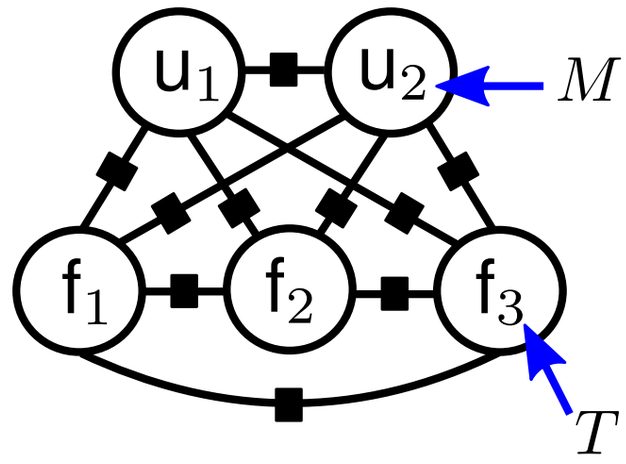
A Unifying Framework for Sparse Gaussian Process Approximation using Power Expectation Propagation
Bui, Yan and Turner, 2016
(VFE, EP, FITC, PITC ...)

FITC: Snelson et al. “Sparse Gaussian Processes using Pseudo-inputs”
PITC: Snelson et al. “Local and global sparse Gaussian process approximations”
EP: Csato and Opper 2002 / Qi et al. "Sparse-posterior Gaussian Processes for general likelihoods.”
VFE: Titsias “Variational Learning of Inducing Variables in Sparse Gaussian Processes”
DTC / PP: Seeger et al. “Fast Forward Selection to Speed Up Sparse Gaussian Process Regression”

FITC

1. Add pseudo points

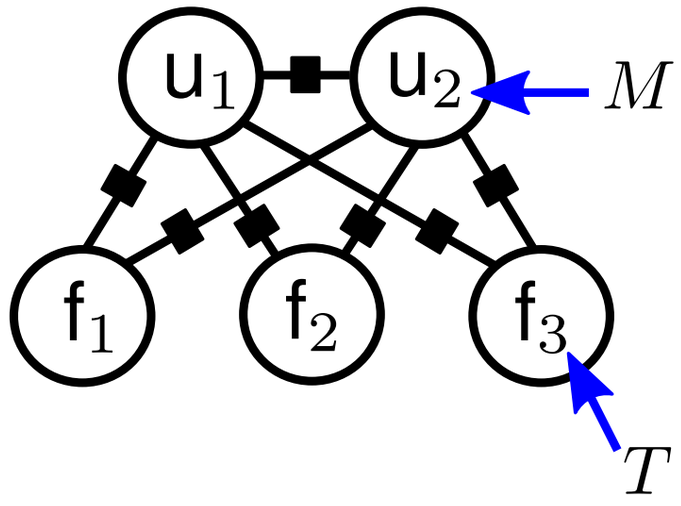
$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix}; \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathsf{K}_{\mathbf{ff}} & \mathsf{K}_{\mathbf{fu}} \\ \mathsf{K}_{\mathbf{uf}} & \mathsf{K}_{\mathbf{uu}} \end{bmatrix}\right)$$



FITC

1. Add pseudo points

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix}; \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathsf{K_{ff}} & \mathsf{K_{fu}} \\ \mathsf{K_{uf}} & \mathsf{K_{uu}} \end{bmatrix}\right)$$

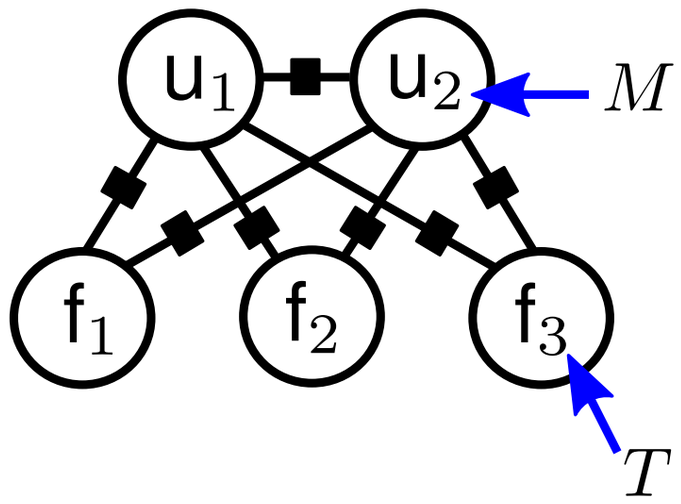


2. drop direct f-f dependencies

FITC

1. Add pseudo points

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix}; \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathsf{K}_{\mathbf{ff}} & \mathsf{K}_{\mathbf{fu}} \\ \mathsf{K}_{\mathbf{uf}} & \mathsf{K}_{\mathbf{uu}} \end{bmatrix}\right)$$



2. drop direct f-f dependencies

3. calibrate model

(e.g. using KL divergence, many choices)

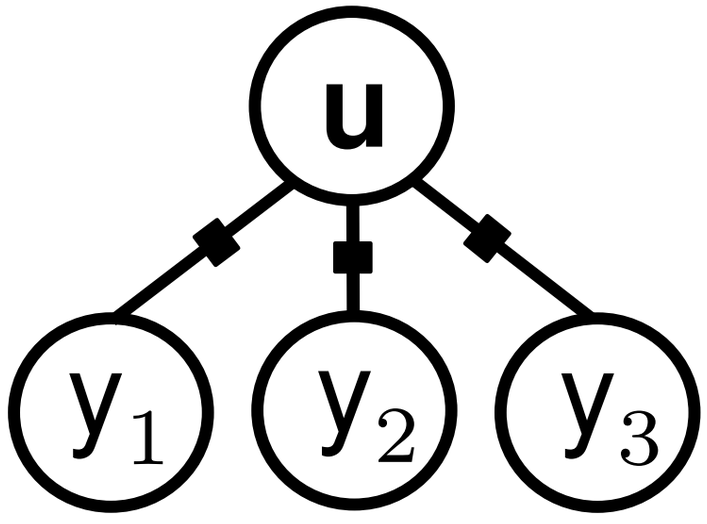
$$\underset{q(\mathbf{u}),\{q(\mathsf{f}_t|\mathbf{u})\}_{t=1}^{T}}{\arg\min} \mathsf{KL}(p(\mathbf{f}, \mathbf{u})||q(\mathbf{u})\prod_{t=1}^{T} q(\mathsf{f}_t|\mathbf{u})) \implies \begin{aligned} q(\mathbf{u}) &= p(\mathbf{u}) \\ q(\mathsf{f}_t|\mathbf{u}) &= p(\mathsf{f}_t|\mathbf{u}) \end{aligned}$$

equal to exact conditionals

FITC

1. Add pseudo points

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix}; \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathsf{K}_{\mathbf{ff}} & \mathsf{K}_{\mathbf{fu}} \\ \mathsf{K}_{\mathbf{uf}} & \mathsf{K}_{\mathbf{uu}} \end{bmatrix}\right)$$



2. drop direct f-f dependencies

3. calibrate model

(e.g. using KL divergence, many choices)

$$\underset{q(\mathbf{u}),\{q(\mathsf{f}_t|\mathbf{u})\}_{t=1}^{T}}{\arg\min} \mathrm{KL}(p(\mathbf{f}, \mathbf{u}) || q(\mathbf{u}) \prod_{t=1}^{T} q(\mathsf{f}_t|\mathbf{u})) \implies \begin{aligned} q(\mathbf{u}) &= p(\mathbf{u}) \\ q(\mathsf{f}_t|\mathbf{u}) &= p(\mathsf{f}_t|\mathbf{u}) \end{aligned}$$

equal to exact conditionals

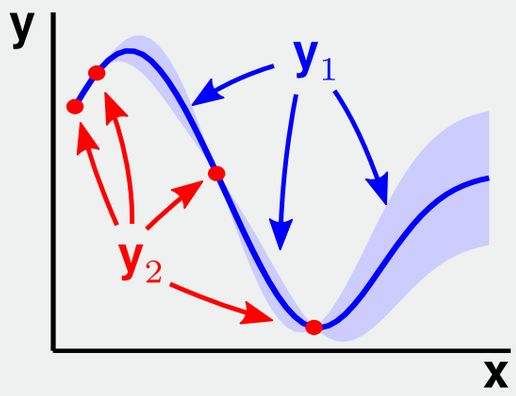
4. Replace p with simpler model

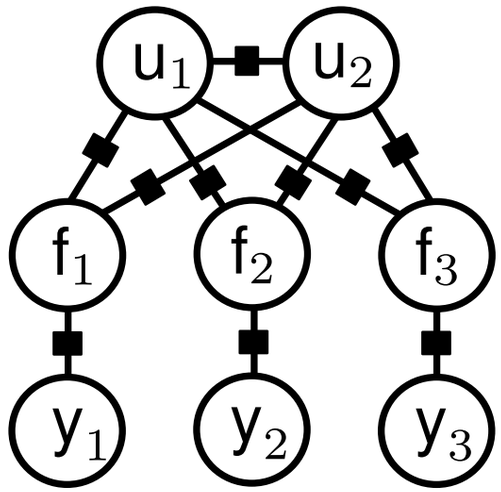
$$q(\mathbf{u}) = p(\mathbf{u}) = \mathcal{N}(\mathbf{u}; 0, \mathsf{K}_{\mathsf{uu}})$$

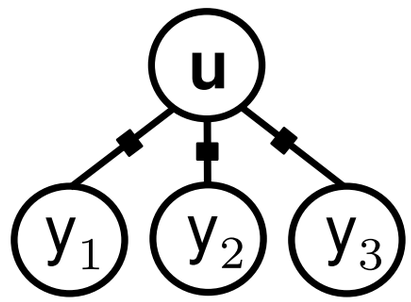
$$q(\mathsf{f}_t|\mathbf{u}) = p(\mathsf{f}_t|\mathbf{u})$$

How do we make predictions?

$$p(\mathbf{y}_1|\mathbf{y}_2) = \mathcal{N}(\mathbf{y}_1; \Sigma_{12}\Sigma_{22}^{-1}\mathbf{y}_2, \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}^{\intercal})$$





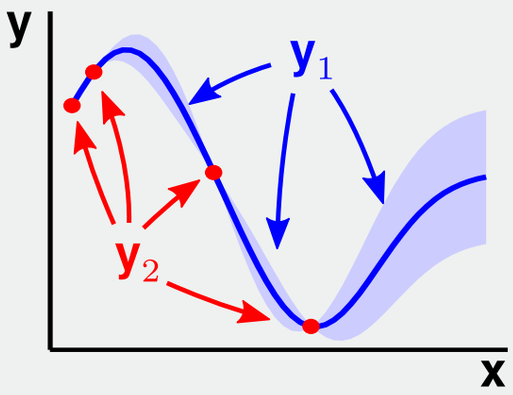


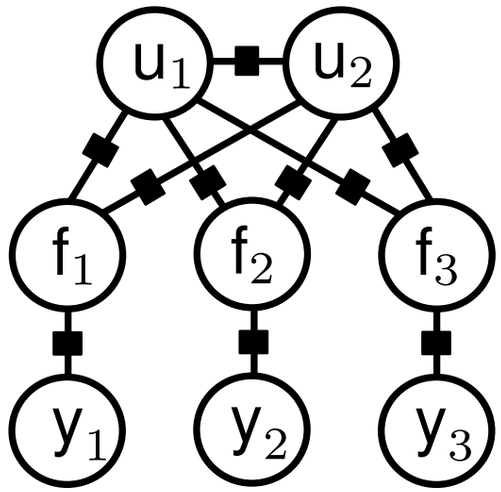
$$q(\mathbf{u}) = p(\mathbf{u}) = \mathcal{N}(\mathbf{u}; 0, \mathsf{K}_{\mathsf{uu}})$$

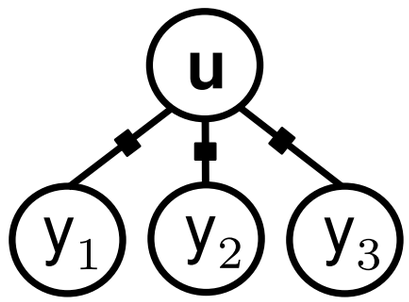
$$q(\mathsf{f}_t|\mathbf{u}) = p(\mathsf{f}_t|\mathbf{u})$$

How do we make predictions?

$$p(\mathbf{y}_1|\mathbf{y}_2) = \mathcal{N}(\mathbf{y}_1; \Sigma_{12}\Sigma_{22}^{-1}\mathbf{y}_2, \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}^\intercal)$$







$$q(\mathsf{f}_t|\mathbf{u}) = p(\mathsf{f}_t|\mathbf{u})$$
$$= \mathcal{N}(\mathsf{f}_t; \mathsf{K}_{\mathsf{f}_t\mathsf{u}}\mathsf{K}_{\mathsf{uu}}^{-1}\mathbf{u}, \underbrace{\mathsf{K}_{\mathsf{f}_t\mathsf{f}_t} - \mathsf{K}_{\mathsf{f}_t\mathsf{u}}\mathsf{K}_{\mathsf{uu}}^{-1}\mathsf{K}_{\mathsf{uf}_t}}_{\mathsf{D}_{tt}})$$

$$q(\mathsf{y}_t|\mathsf{f}_t) = p(\mathsf{y}_t|\mathsf{f}_t) = \mathcal{N}(\mathsf{y}_t; \mathsf{f}_t, \sigma_{\mathsf{y}}^2)$$

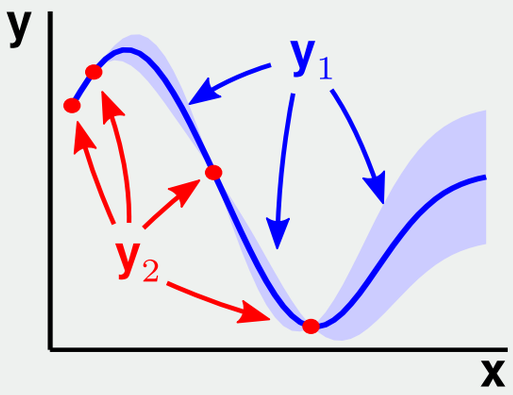
cost of computing likelihood is $\mathcal{O}(TM^2)$

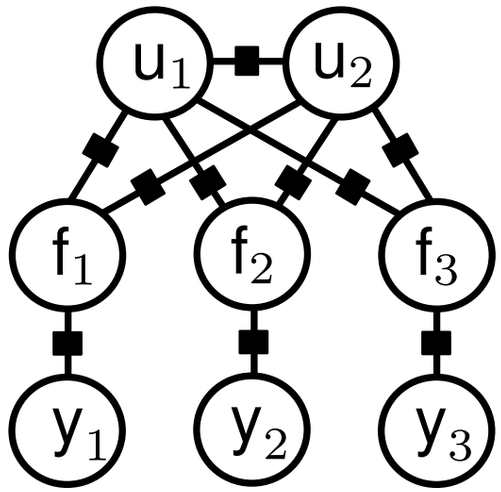
$$q(\mathbf{u}) = p(\mathbf{u}) = \mathcal{N}(\mathbf{u}; 0, \mathsf{K}_{\mathsf{uu}})$$

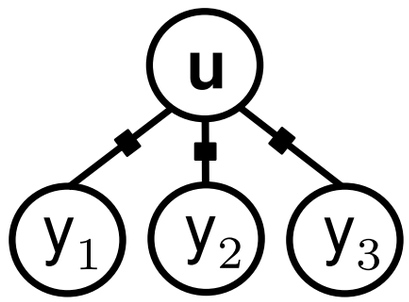
$$q(\mathsf{f}_t|\mathbf{u}) = p(\mathsf{f}_t|\mathbf{u})$$

How do we make predictions?

$$p(\mathbf{y}_1|\mathbf{y}_2) = \mathcal{N}(\mathbf{y}_1; \Sigma_{12}\Sigma_{22}^{-1}\mathbf{y}_2, \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}^\intercal)$$







$$q(\mathsf{f}_t|\mathbf{u}) = p(\mathsf{f}_t|\mathbf{u})$$
$$= \mathcal{N}(\mathsf{f}_t; \mathsf{K}_{\mathsf{f}_t\mathsf{u}}\mathsf{K}_{\mathsf{uu}}^{-1}\mathbf{u}, \underbrace{\mathsf{K}_{\mathsf{f}_t\mathsf{f}_t} - \mathsf{K}_{\mathsf{f}_t\mathsf{u}}\mathsf{K}_{\mathsf{uu}}^{-1}\mathsf{K}_{\mathsf{uf}_t}}_{\mathsf{D}_{tt}})$$

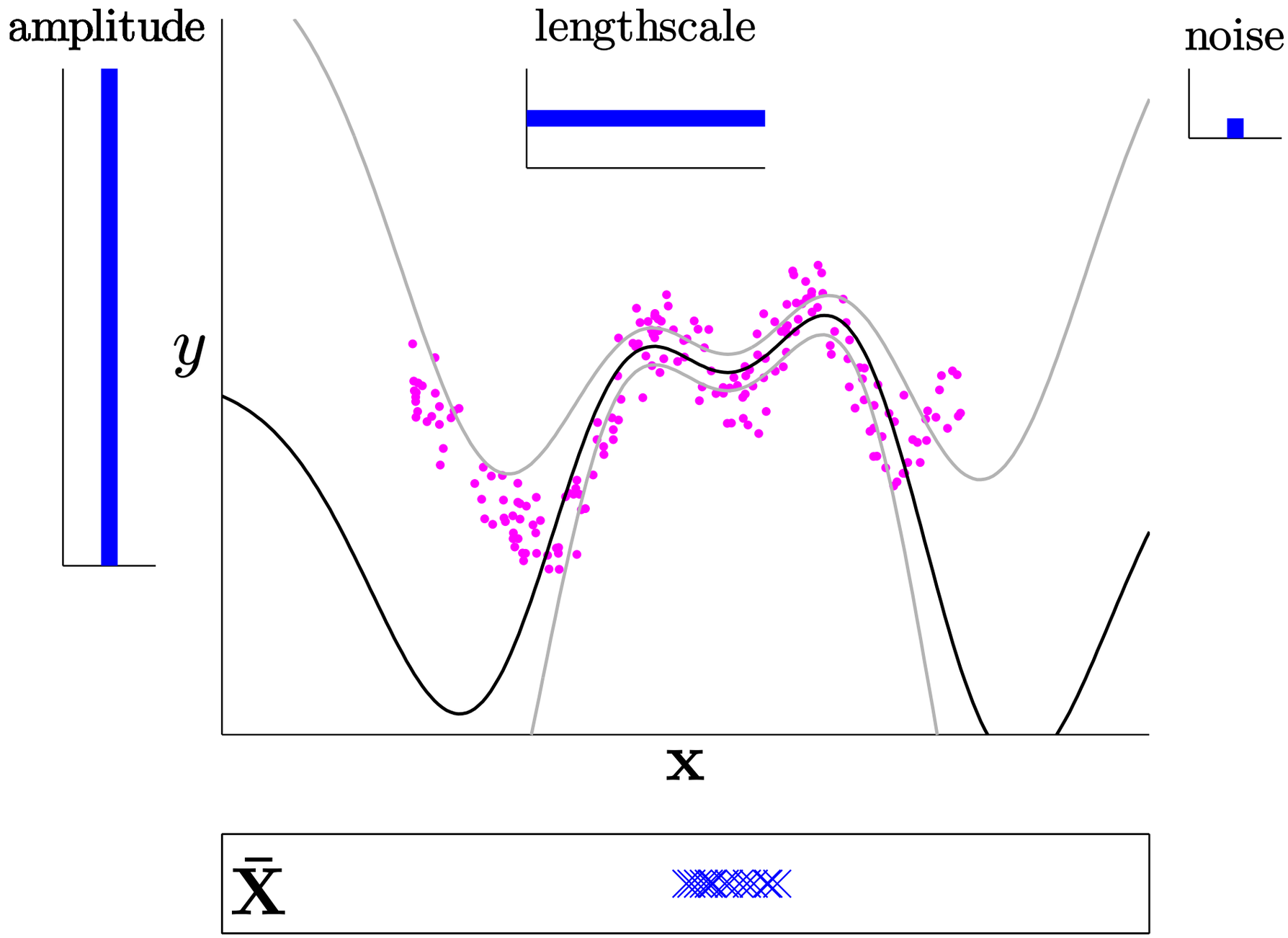
$$q(\mathsf{y}_t|\mathsf{f}_t) = p(\mathsf{y}_t|\mathsf{f}_t) = \mathcal{N}(\mathsf{y}_t; \mathsf{f}_t, \sigma_{\mathsf{y}}^2)$$

cost of computing likelihood is $\mathcal{O}(TM^2)$

cost of computing likelihood is $\mathcal{O}(TM^2)$

$$p(\mathbf{y}_t|\theta) = \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathsf{K}_{\mathsf{fu}}\mathsf{K}_{\mathsf{uu}}^{-1}\mathsf{K}_{\mathsf{uu}}\mathsf{K}_{\mathsf{uu}}^{-1}\mathsf{K}_{\mathsf{uf}} + \mathsf{D} + \sigma_{\mathsf{y}}^2\mathsf{I})$$

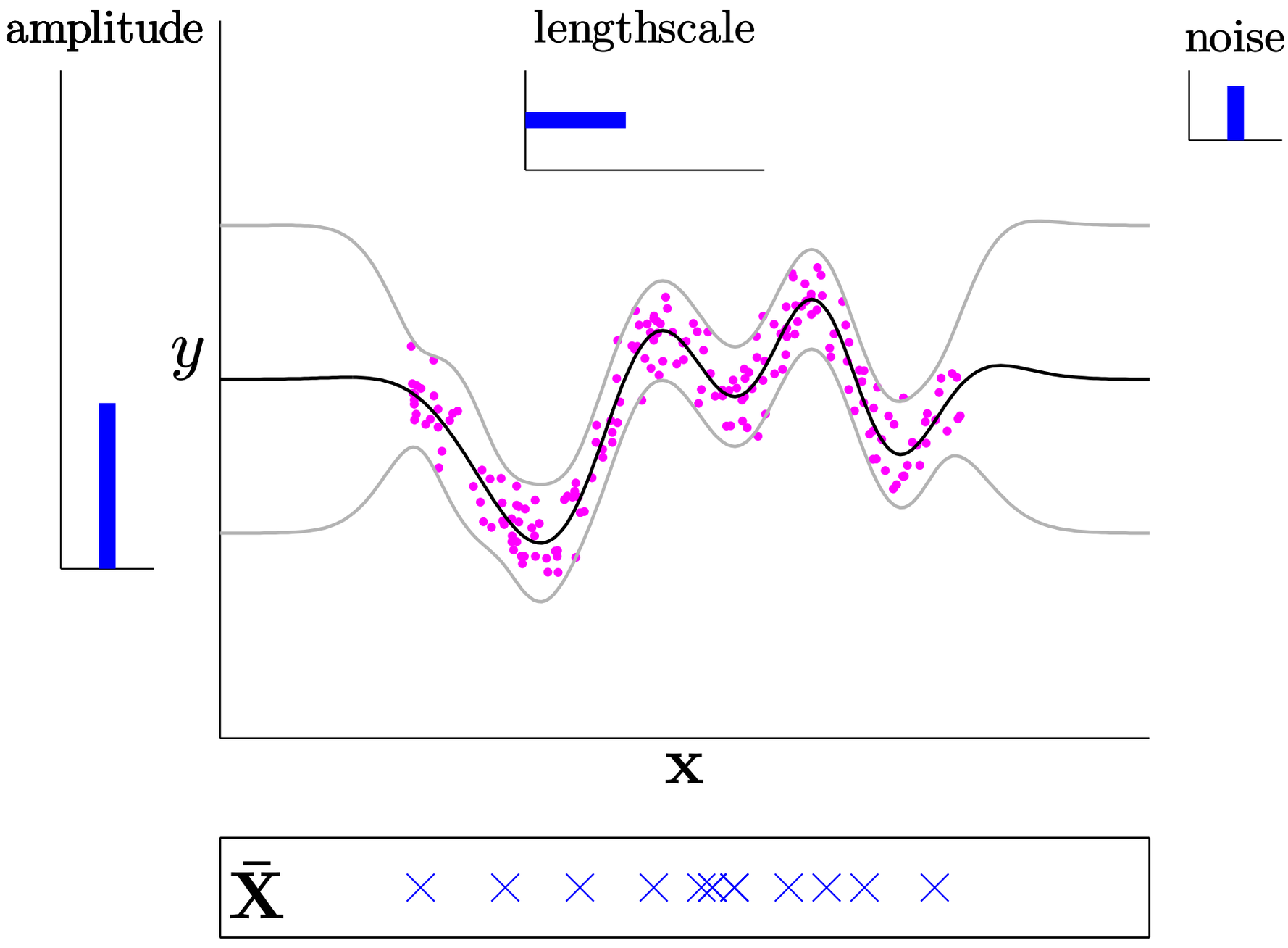
FITC: Demo (Snelson)



Initialize adversarially:

amplitude and lengthscale too big
noise too small
pseudo-inputs bunched up

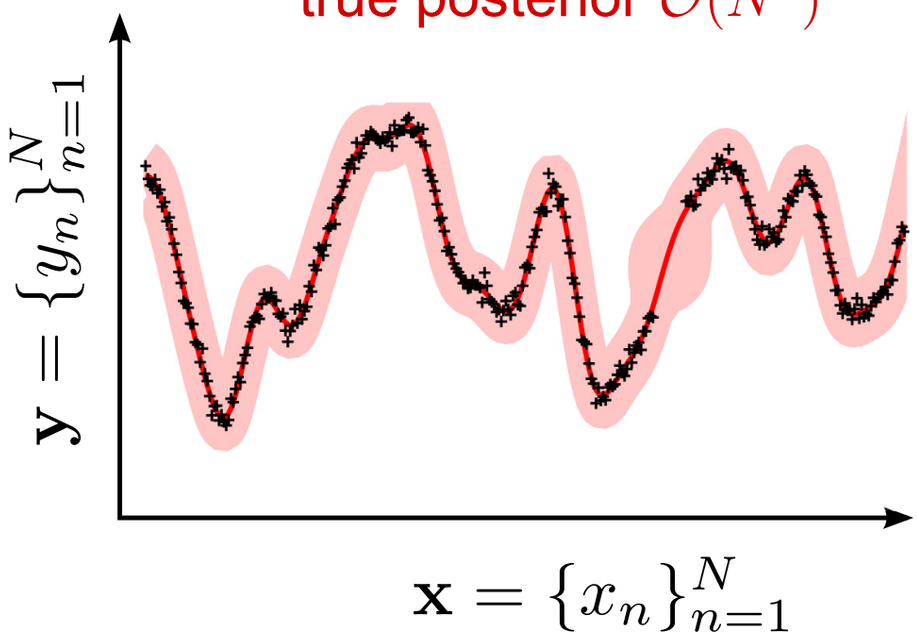
FITC: Demo (Snelson)



Pseudo-inputs and hyperparameters optimized

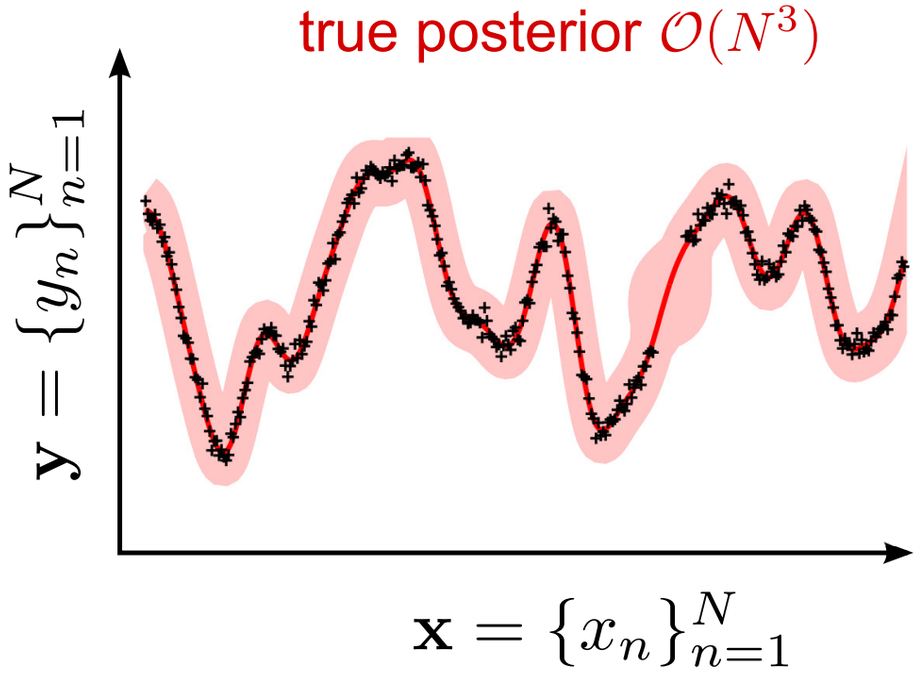
EP pseudo-point approximation

$$p^*(f) = \; p(f, \mathbf{y}|\mathbf{x}, \theta)$$



EP pseudo-point approximation

$$
\begin{aligned}
p^*(f) = \; & p(f, \mathbf{y}|\mathbf{x}, \theta) \\
& = p(f|\theta) \prod_{n=1}^{N} p(y_n|f, x_n, \theta)
\end{aligned}
$$

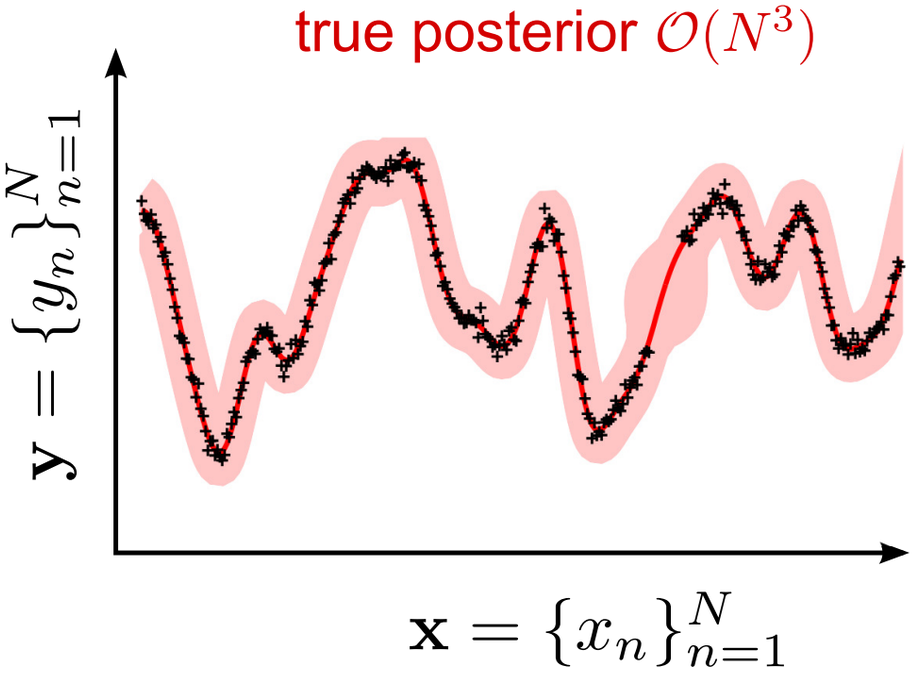


EP pseudo-point approximation

$$
\begin{aligned}
p^*(f) = \; & p(f, \mathbf{y}|\mathbf{x}, \theta) \\
& = p(f|\theta) \prod_{n=1}^{N} p(y_n|f, x_n, \theta) \\
& = p(\mathbf{y}|\mathbf{x}, \theta) \; p(f|\mathbf{y}, \mathbf{x}, \theta)
\end{aligned}
$$

marginal likelihood

posterior

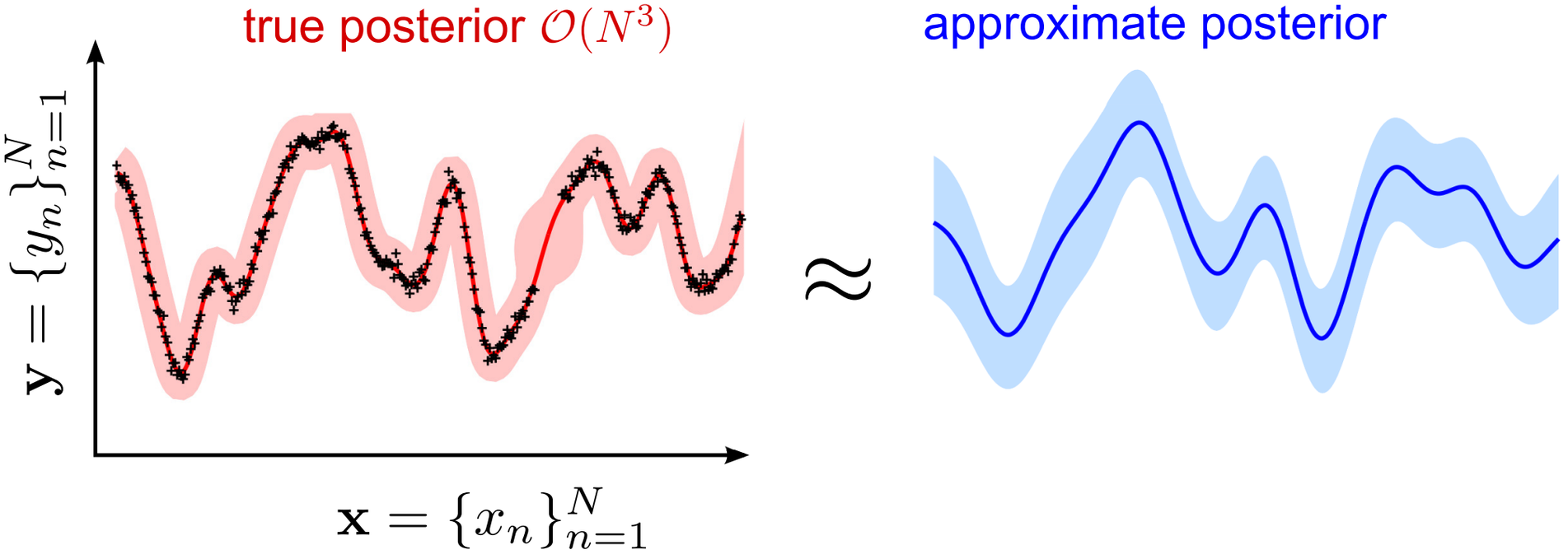


EP pseudo-point approximation

$$p^*(f) = p(f, \mathbf{y}|\mathbf{x}, \theta)$$
$$= p(f|\theta) \prod_{n=1}^{N} p(y_n|f, x_n, \theta) \qquad q^*(f) = p(f|\theta) \prod_{n=1}^{N} t_n(f)$$
$$= p(\mathbf{y}|\mathbf{x}, \theta)\ p(f|\mathbf{y}, \mathbf{x}, \theta)$$

marginal likelihood

posterior



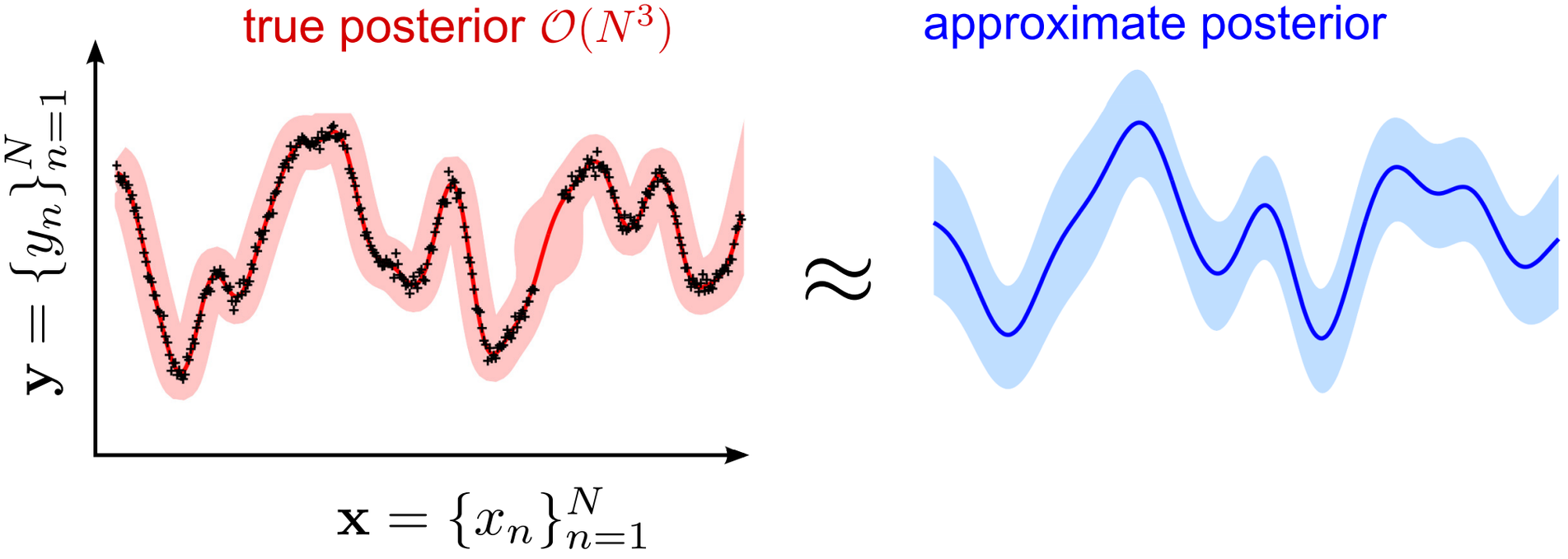
EP pseudo-point approximation

$$p^*(f) = p(f, \mathbf{y}|\mathbf{x}, \theta)$$
$$= p(f|\theta) \prod_{n=1}^{N} p(y_n|f, x_n, \theta)$$
$$= p(\mathbf{y}|\mathbf{x}, \theta)\ p(f|\mathbf{y}, \mathbf{x}, \theta)$$

marginal likelihood

posterior

$$q^*(f) = p(f|\theta) \prod_{n=1}^{N} t_n(f)$$
$$= Z_{\mathrm{EP}}\ q(f)$$



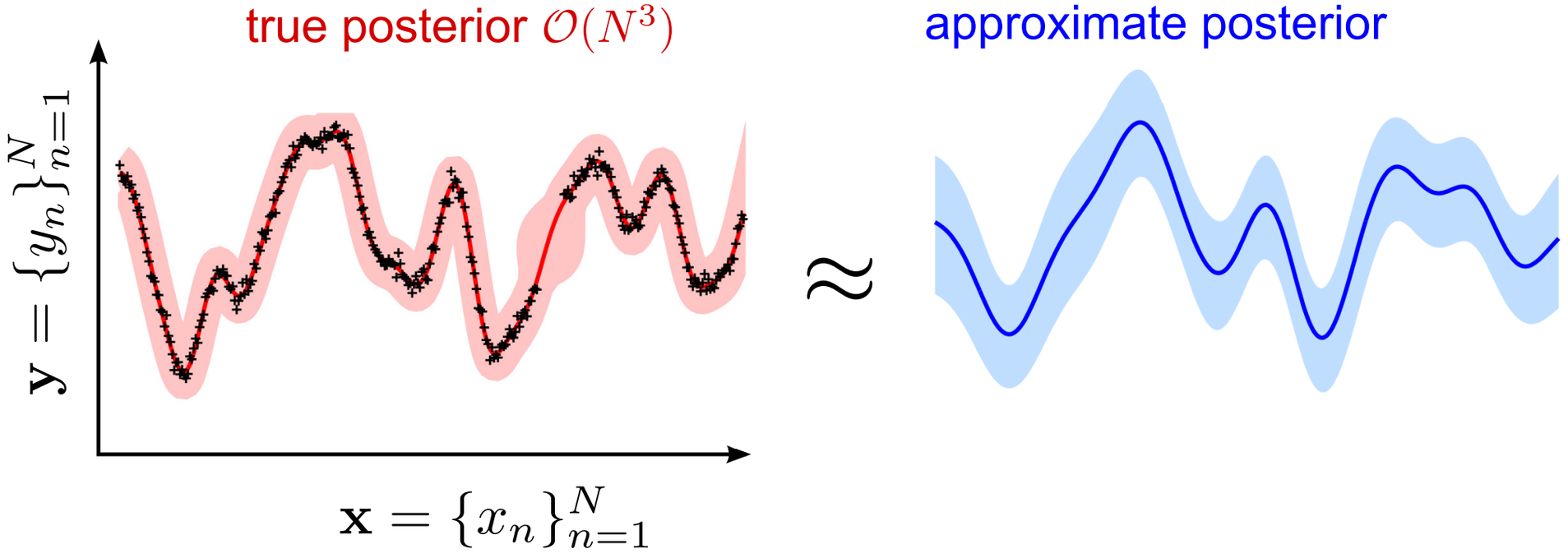
EP pseudo-point approximation

$$p^*(f) = p(f, \mathbf{y}|\mathbf{x}, \theta)$$
$$= p(f|\theta) \prod_{n=1}^{N} p(y_n|f, x_n, \theta)$$
$$= p(\mathbf{y}|\mathbf{x}, \theta)\ p(f|\mathbf{y}, \mathbf{x}, \theta)$$

marginal likelihood — posterior

$$q^*(f) = p(f|\theta) \prod_{n=1}^{N} t_n(f)$$
$$= Z_{\mathrm{EP}}\ q(f)$$

$$t_n(f) = \mathcal{N}(\mathbf{u}; \mu_n, \Sigma_n)$$
$$\dim(\mathbf{u}) = M \quad f = \{\mathbf{u}, f_{\neq \mathbf{u}}\}$$



EP pseudo-point approximation

$$p^*(f) = p(f, \mathbf{y}|\mathbf{x}, \theta)$$
$$= p(f|\theta) \prod_{n=1}^{N} p(y_n|f, x_n, \theta)$$
$$= p(\mathbf{y}|\mathbf{x}, \theta)\ p(f|\mathbf{y}, \mathbf{x}, \theta)$$

marginal likelihood

posterior

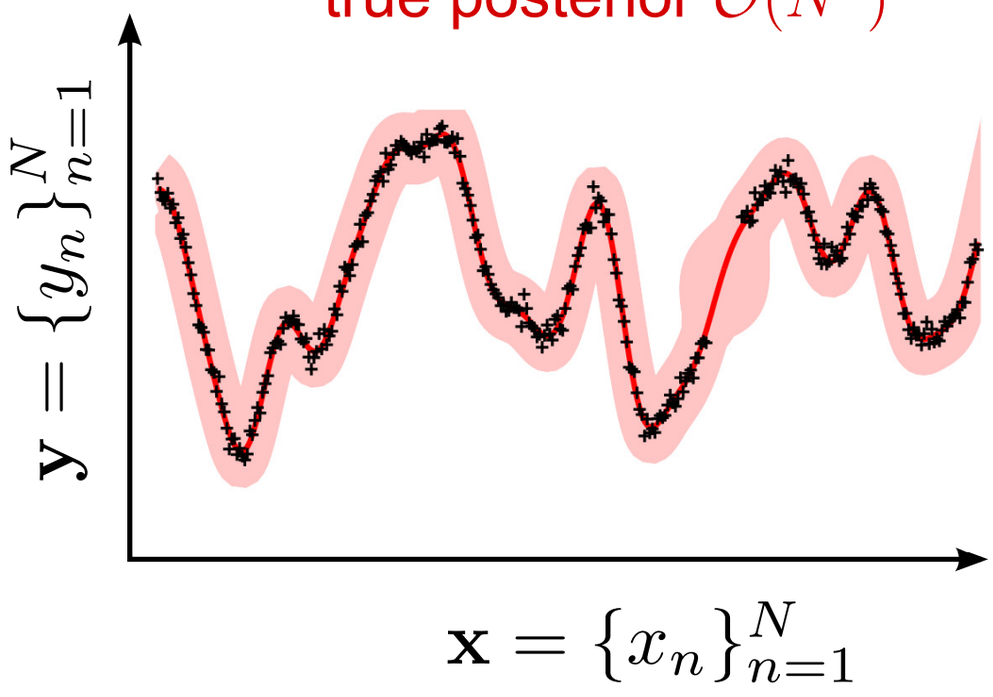
$$q^*(f) = p(f|\theta)p(\tilde{\mathbf{y}}|\mathbf{u}, \tilde{\Sigma})$$
$$= p(f|\theta) \prod_{n=1}^{N} t_n(f)$$
$$= Z_{\text{EP}}\ q(f)$$

exact joint of new GP regression model

$$t_n(f) = \mathcal{N}(\mathbf{u}; \mu_n, \Sigma_n)$$

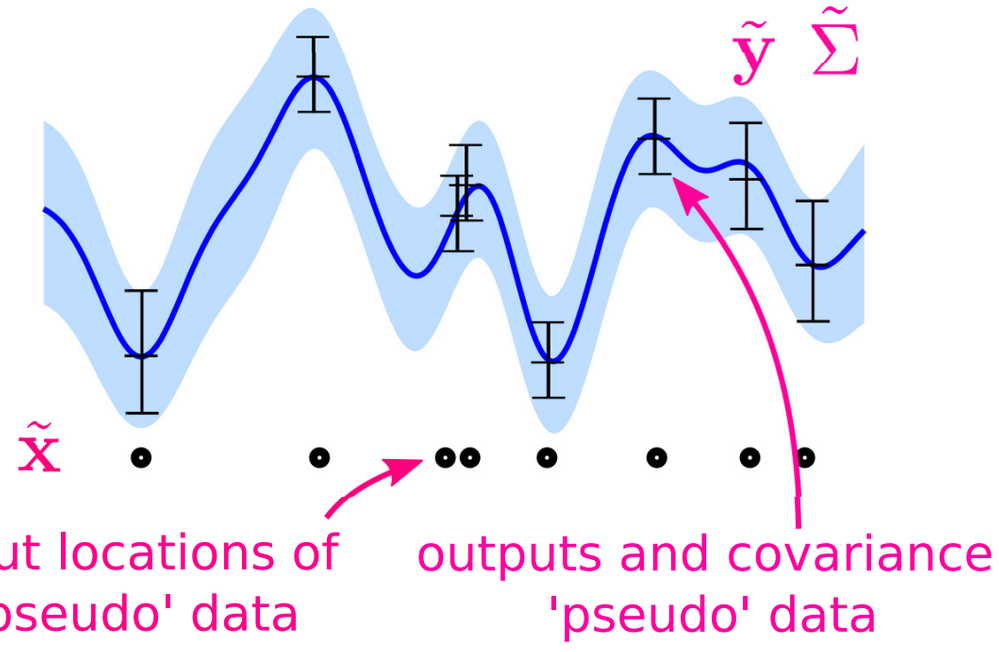
$$\dim(\mathbf{u}) = M \quad f = \{\mathbf{u}, f_{\neq \mathbf{u}}\}$$

true posterior $\mathcal{O}(N^3)$



$\approx$

approximate posterior $\mathcal{O}(NM^2)$



EP algorithm

1. remove

$$q^{\setminus n}(f) = \frac{q^*(f)}{t_n(\mathbf{u})}$$

cavity

take out one pseudo-observation likelihood

EP algorithm

1. remove

$$q^{\setminus n}(f) = \frac{q^*(f)}{t_n(\mathbf{u})}$$

cavity

take out one pseudo-observation likelihood

2. include

$$p_n^{\text{tilt}}(f) = q^{\setminus n}(f)p(y_n|f, x_n, \theta)$$

tilted

add in one true observation likelihood

EP algorithm

1. remove $q^{\setminus n}(f) = \frac{q^*(f)}{t_n(\mathbf{u})}$ — take out one pseudo-observation likelihood

cavity

2. include $p_n^{\text{tilt}}(f) = q^{\setminus n}(f)p(y_n|f, x_n, \theta)$ — add in one true observation likelihood

tilted

3. project $q^*(f) = \underset{q^*(f)}{\operatorname{argmin}} \mathrm{KL}\left[p_n^{\text{tilt}}(f)||q^*(f)\right]$ — project onto approximating family

KL between unnormalised stochastic processes

1. minimum: moments matched at pseudo-inputs $\mathcal{O}(NM^2)$
2. Gaussian regression: matches moments everywhere

EP algorithm

1. remove $q^{\backslash n}(f) = \frac{q^*(f)}{t_n(\mathbf{u})}$ — take out one pseudo-observation likelihood

cavity

2. include $p_n^{\text{tilt}}(f) = q^{\backslash n}(f) p(y_n|f, x_n, \theta)$ — add in one true observation likelihood

tilted

3. project $q^*(f) = \underset{q^*(f)}{\operatorname{argmin}} \operatorname{KL}\left[p_n^{\text{tilt}}(f) || q^*(f)\right]$ — project onto approximating family

KL between unnormalised stochastic processes

1. minimum: moments matched at pseudo-inputs $\mathcal{O}(NM^2)$
2. Gaussian regression: matches moments everywhere

4. update
$$t_n(\mathbf{u}) = \frac{q^*(f)}{q^{\backslash n}(f)} = z_n \mathcal{N}(\mathrm{K}_{f_n \mathbf{u}} \mathrm{K}_{\mathbf{uu}}^{-1} \mathbf{u}; g_n, v_n)$$
update pseudo-observation likelihood

rank 1

Further reading:

Approximate inference in GPs:

- Sparse Online Gaussian Processes, Csato and Opper, Neural Computation, 2002
- A Unifying View of Sparse Approximate Gaussian Process Regression, Quinonero-Candela and Rasm JMLR, 2005
- Variational Learning of Inducing Variables in Sparse Gaussian Processes Titsias, AIStats, 2009
- On Sparse Variational Methods and the Kullback-Leibler Divergence between Stochastic Processes, Matthews et al., ICML 2016
- A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation, Bui et al., JMLR 2017
- Streaming Sparse Gaussian Process Approximations, Bui et al., NIPS 2017
- Efficient Deterministic Approximate Bayesian Inference for Gaussian Process Models , Bui, thesis, 20

Deep Gaussian Processes:

- Deep Gaussian Processes for Regression using Approximate Expectation Propagation, Bui et al., ICM
- Doubly Stochastic Variational Inference for Deep Gaussian Processes Salimbeni and Deisenroth, NIPS

Gaussian Process Models

- Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data
- Lawrence, NIPS, 2005
  Local Distance Preservation in the GP-LVM through Back Constraints, Lawrence and
- Quinonero-Candela, ICML 2006

The Automatic Statistician

- The Automatic Statistician, Ghahramani et al., (website link)

GPs for Reinforcement Learning and Control

- PILCO: A Model-Based and Data-Efficient Approach to Policy Search, Deisenroth and
- Rasmussen, ICML 2011

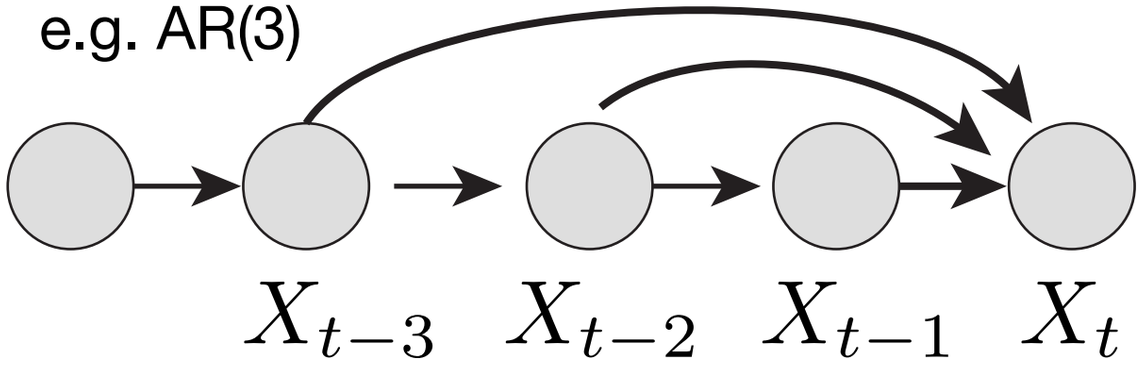
GPs and Neural Networks

- Bayesian Learning for Neural Networks Neal, 1996
- Gaussian Process Behaviour in Wide Deep Neural Networks, Matthews et al., arXiv, 2018

How do we use these to model time series?

How do we use these to model time series?

ARIMA: directly model data statistics



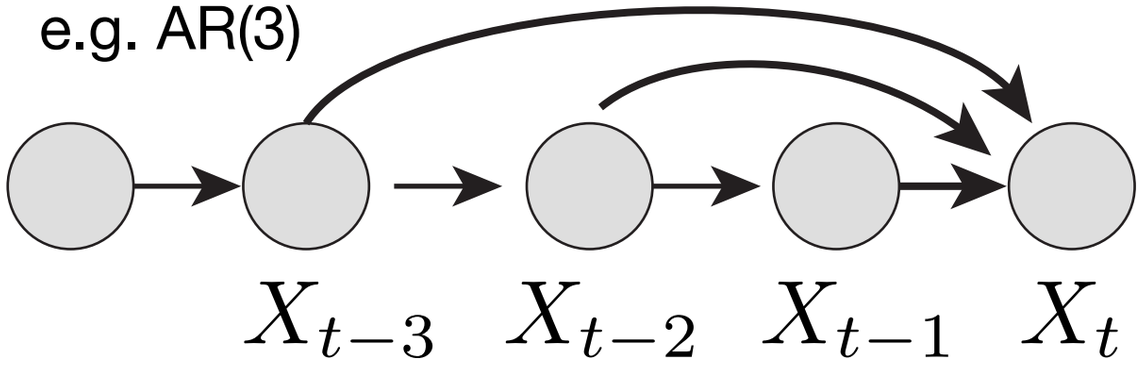
simple dependencies

linear prediction

GP: complex functional dependencies

How do we use these to model time series?

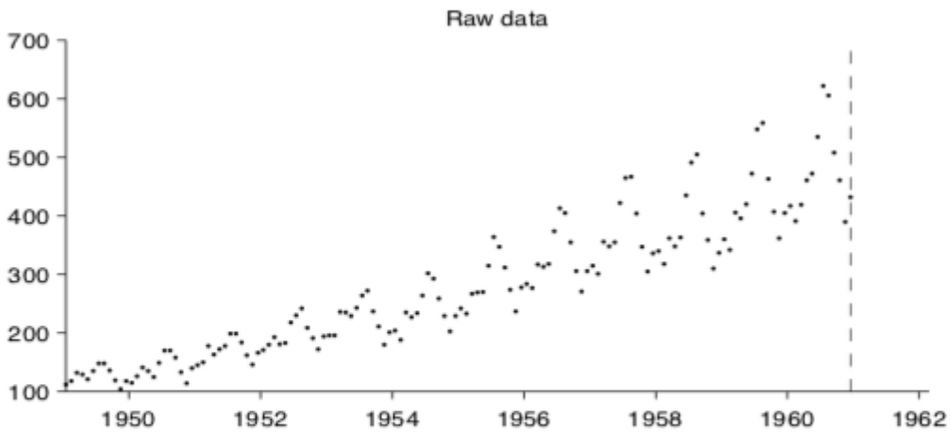
ARIMA: directly model data statistics

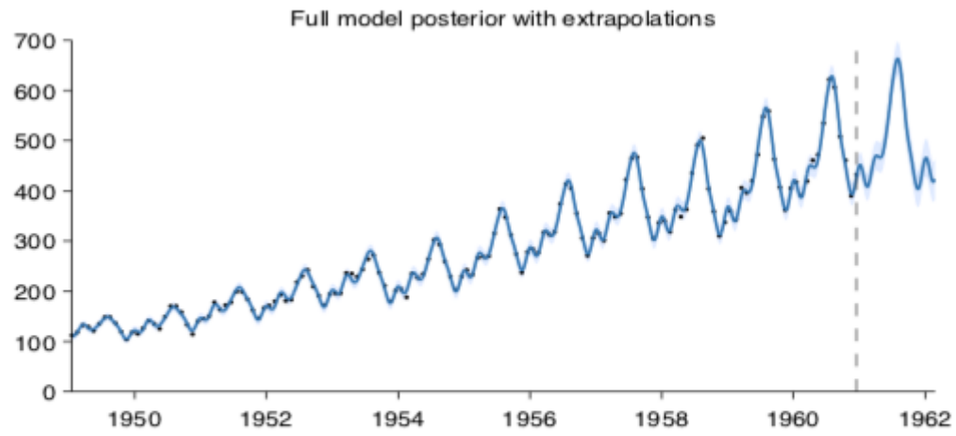


simple dependencies

linear prediction

GP: complex functional dependencies





Deep Kalman Filter *(Krishnan et al, 2016)*

Transition distribution

$$z_t|z_{t-1} \sim \mathcal{N}(F_\theta^\mu(z_{t-1}), F_\theta^\Sigma(z_{t-1}))$$

Emission distribution

$$x_t|z_t \sim \mathcal{N}(G_\theta^\mu(z_t), G_\theta^\Sigma(z_t))$$

Approximate posterior distribution

$$z_t|x_{1:T} \sim \mathcal{N}(R_\theta^\mu(x_{1:T}), R_\theta^\Sigma(x_{1:T}))$$

Deep Kalman Filter *(Krishnan et al, 2016)*

Transition distribution

$$z_t|z_{t-1} \sim \mathcal{N}(F_\theta^\mu(z_{t-1}), F_\theta^\Sigma(z_{t-1}))$$

Emission distribution

$$x_t|z_t \sim \mathcal{N}(G_\theta^\mu(z_t), G_\theta^\Sigma(z_t))$$

Feedforward neural networks

Approximate posterior distribution

$$z_t|x_{1:T} \sim \mathcal{N}(R_\theta^\mu(x_{1:T}), R_\theta^\Sigma(x_{1:T}))$$

Deep Kalman Filter *(Krishnan et al, 2016)*

Transition distribution

$$z_t|z_{t-1} \sim \mathcal{N}(F_\theta^\mu(z_{t-1}), F_\theta^\Sigma(z_{t-1}))$$

Emission distribution

$$x_t|z_t \sim \mathcal{N}(G_\theta^\mu(z_t), G_\theta^\Sigma(z_t))$$

Approximate posterior distribution

$$z_t|x_{1:T} \sim \mathcal{N}(R_\theta^\mu(x_{1:T}), R_\theta^\Sigma(x_{1:T}))$$

Feedforward neural networks

Bidirectional recurrent networks

Deep Kalman Filter *(Krishnan et al, 2016)*

Transition distribution

$$z_t|z_{t-1} \sim \mathcal{N}(F_\theta^\mu(z_{t-1}), F_\theta^\Sigma(z_{t-1}))$$

Emission distribution

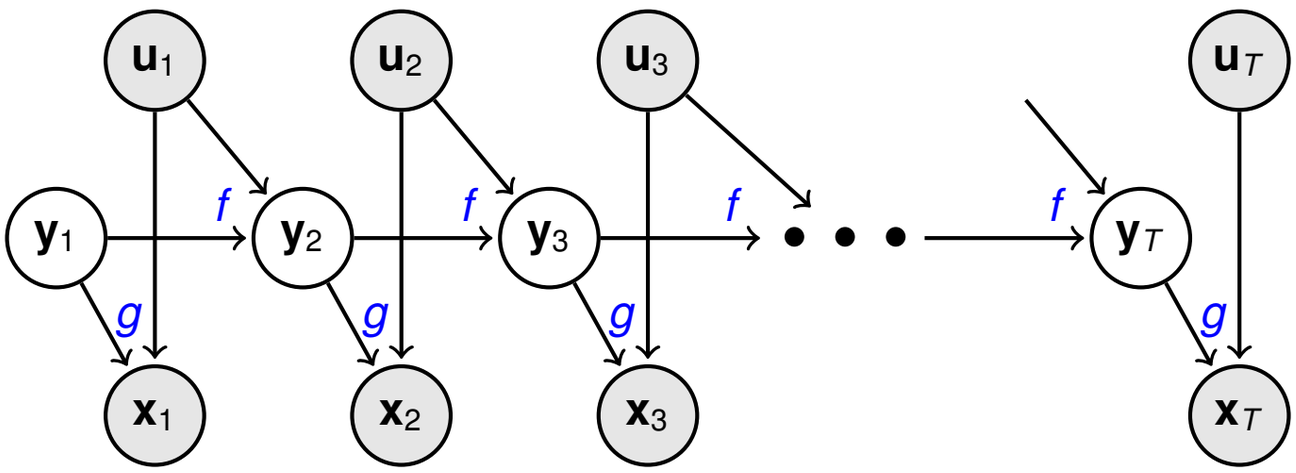
$$x_t|z_t \sim \mathcal{N}(G_\theta^\mu(z_t), G_\theta^\Sigma(z_t))$$

Approximate posterior distribution

$$z_t|x_{1:T} \sim \mathcal{N}(R_\theta^\mu(x_{1:T}), R_\theta^\Sigma(x_{1:T}))$$

Feedforward neural networks

Bidirectional recurrent networks



$$\mathbf{y}_{t+1} = f(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{w}_t$$

$$\mathbf{x}_t = g(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{v}_t$$

$\mathbf{w}_t, \mathbf{v}_t$ usually assumed Gaussian.

Deep Kalman Filter *(Krishnan et al, 2016)*

Transition distribution

$$z_t|z_{t-1} \sim \mathcal{N}(F_\theta^\mu(z_{t-1}), F_\theta^\Sigma(z_{t-1}))$$

Emission distribution

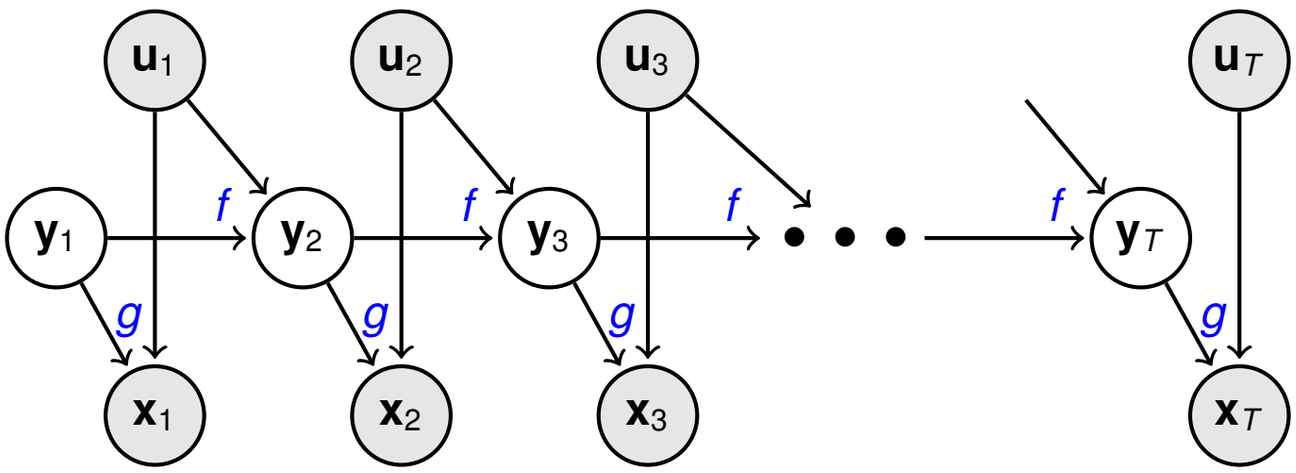
$$x_t|z_t \sim \mathcal{N}(G_\theta^\mu(z_t), G_\theta^\Sigma(z_t))$$

Approximate posterior distribution

$$z_t|x_{1:T} \sim \mathcal{N}(R_\theta^\mu(x_{1:T}), R_\theta^\Sigma(x_{1:T}))$$

Feedforward neural networks

Bidirectional recurrent networks

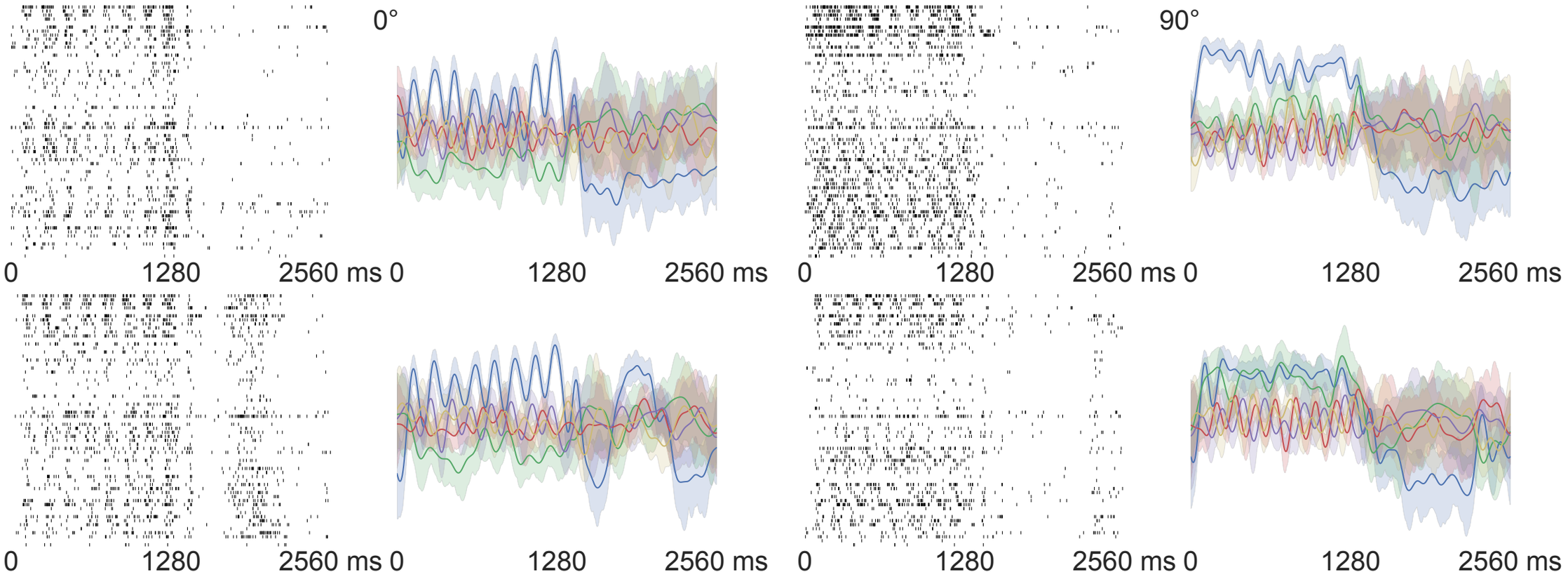


$$\mathbf{y}_{t+1} = f(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{w}_t$$

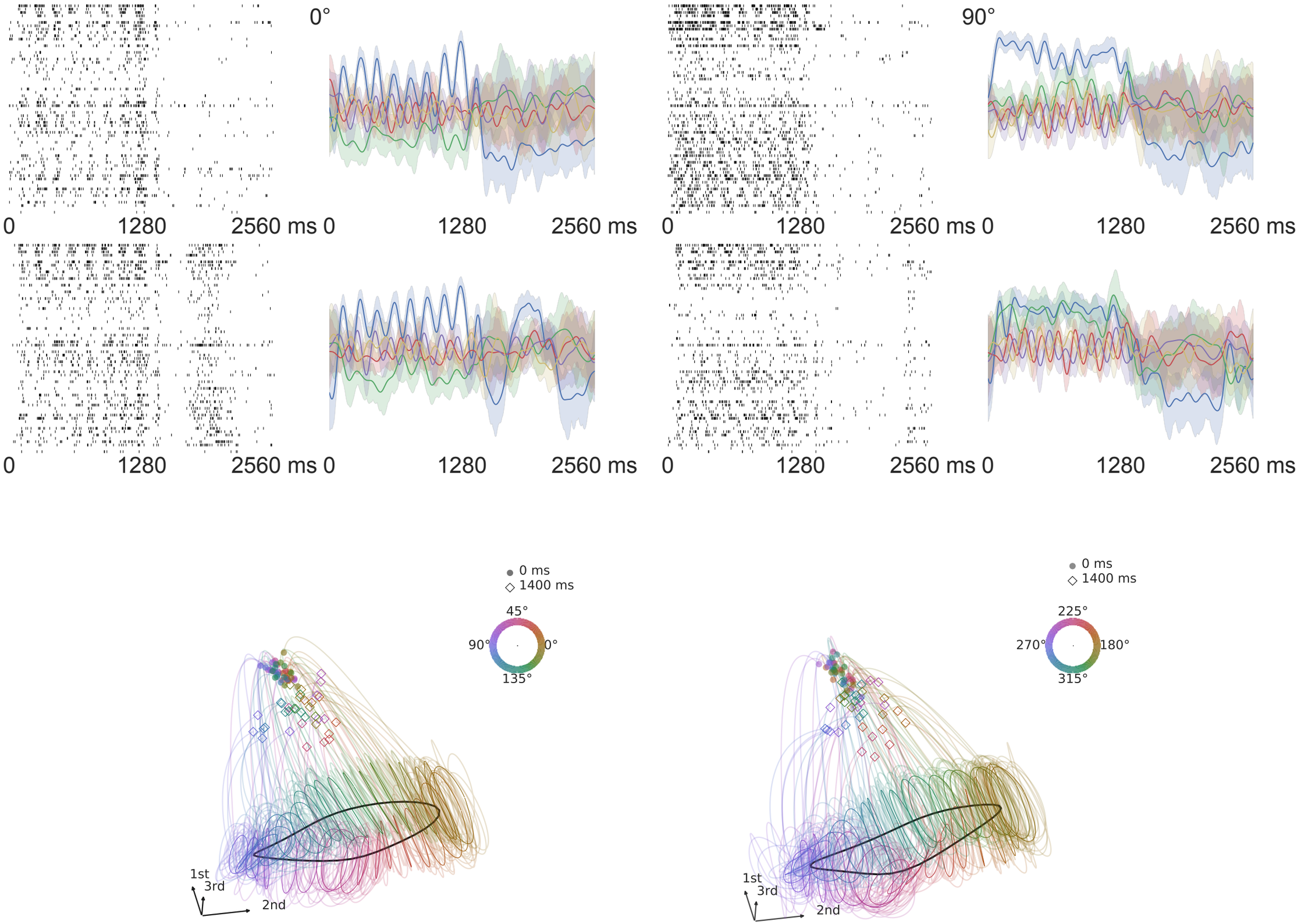
$$\mathbf{x}_t = g(\mathbf{y}_t, \mathbf{u}_t) + \mathbf{v}_t$$

$\mathbf{w}_t, \mathbf{v}_t$ usually assumed Gaussian.

GP priors for f,g

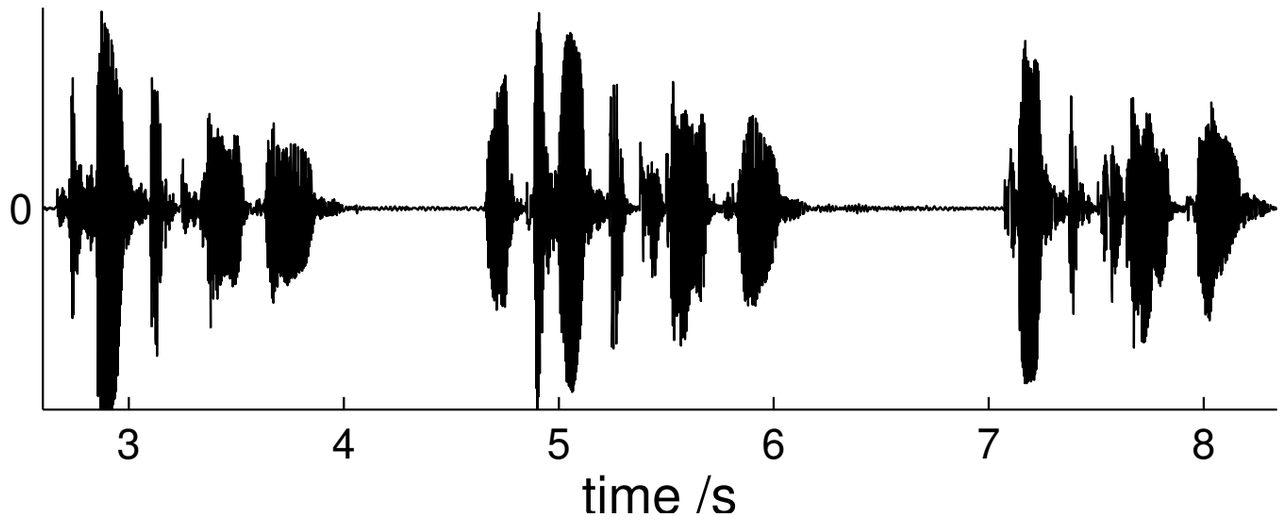


Variational Latent Gaussian Process for Recovering Single-Trial Dynamics from Population Spike Trains , Zhao and Park , 2016

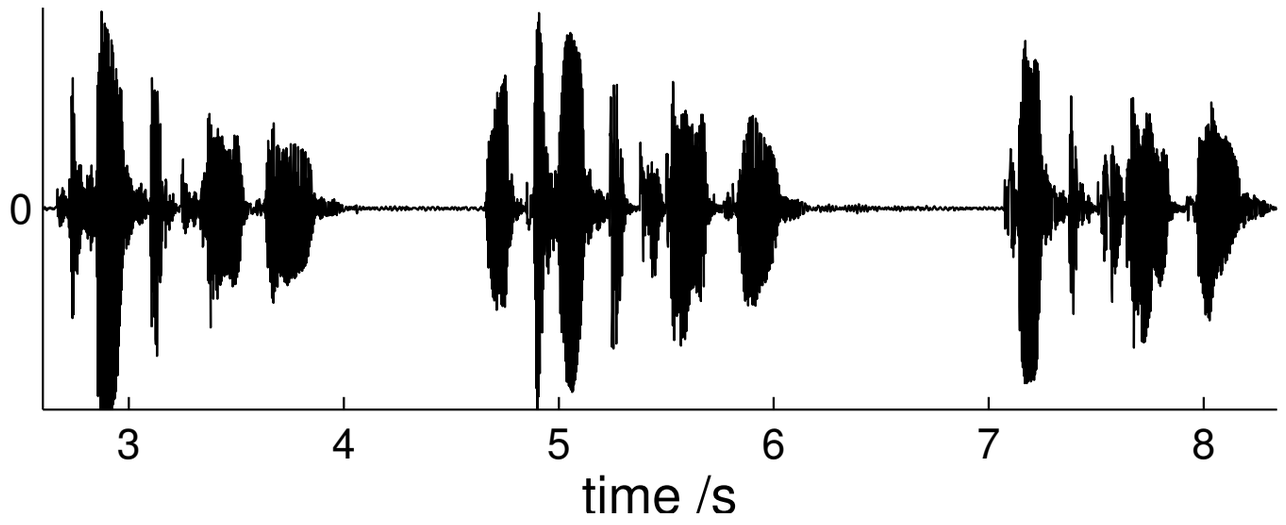


Variational Latent Gaussian Process for Recovering Single-Trial Dynamics from Population Spike Trains , Zhao and Park , 2016

audio
time-series
data



audio
time-series
data



Generative model

$\mathsf{y}(\mathsf{t}) = f(\mathsf{t}) + \epsilon\sigma_{\mathsf{y}}$ (independent Gaussian noise)

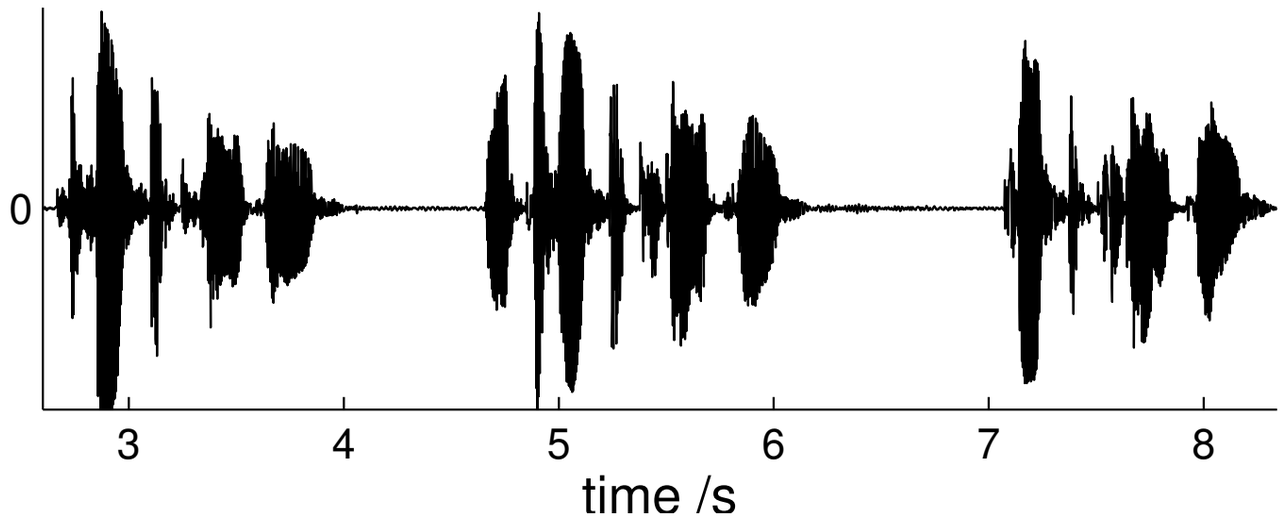
$p(\epsilon) = \mathcal{N}(0, 1)$

place GP prior over the non-linear function

$p(f(\mathsf{t})|\theta) = \mathcal{GP}(0, \mathsf{K}(\mathsf{t}, \mathsf{t}'))$

$\mathsf{K}(\mathsf{t}, \mathsf{t}') = \sigma^2 \cos(\omega(\mathsf{t} - \mathsf{t}')) \exp\left(-\frac{1}{2l^2}(\mathsf{t} - \mathsf{t}')^2\right)$

audio
time-series
data



Generative model

$$\mathsf{y}(\mathsf{t}) = f(\mathsf{t}) + \epsilon\sigma_{\mathsf{y}} \quad \text{(independent Gaussian noise)}$$

$$p(\epsilon) = \mathcal{N}(0, 1)$$

place GP prior over the non-linear function

$$p(f(\mathsf{t})|\theta) = \mathcal{GP}(0, \mathsf{K}(\mathsf{t}, \mathsf{t}'))$$

$$\mathsf{K}(\mathsf{t}, \mathsf{t}') = \sigma^2 \cos(\omega(\mathsf{t} - \mathsf{t}')) \exp\left(-\frac{1}{2l^2}(\mathsf{t} - \mathsf{t}')^2\right)$$

Also: model envelopes in spectral domain…