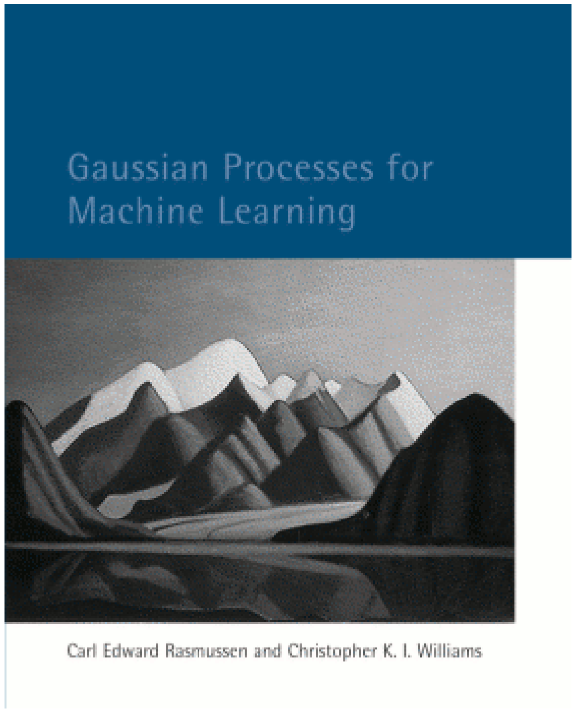
Tutorial: Gaussian process models for machine learning

Ed Snelson (snelson@gatsby.ucl.ac.uk)

Gatsby Computational Neuroscience Unit, UCL

26th October 2006

Info


Gaussian Processes for Machine Learning
Carl Edward Rasmussen and Christopher K. I. Williams

The GP book: Rasmussen and Williams, 2006

Basic GP (Matlab) code available:

http://www.gaussianprocess.org/gpml/

Gaussian process history

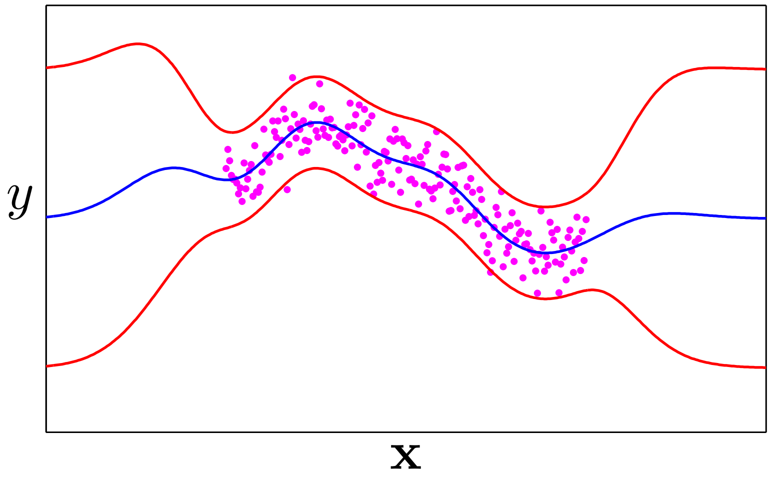
Prediction with GPs:

- Time series: Wiener, Kolmogorov 1940's
- Geostatistics: *kriging* 1970's — naturally only two or three dimensional input spaces
- Spatial statistics in general: see Cressie [1993] for overview
- General regression: O'Hagan [1978]
- Computer experiments (noise free): Sacks et al. [1989]
- Machine learning: Williams and Rasmussen [1996], Neal [1996]

Nonlinear regression

Consider the problem of nonlinear regression:

You want to learn a function f with error bars from data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$



A Gaussian process is a prior over functions $p(f)$ which can be used for Bayesian regression:

$$p(f|\mathcal{D}) = \frac{p(f)p(\mathcal{D}|f)}{p(\mathcal{D})}$$

What is a Gaussian process?

- Continuous stochastic process — random functions — a set of random variables indexed by a continuous variable: $f(x)$
- Set of 'inputs' $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$; corresponding set of random function variables $\mathbf{f} = \{f_1, f_2, \ldots, f_N\}$
- GP: Any set of function variables $\{f_n\}_{n=1}^N$ has joint (zero mean) Gaussian distribution:

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

- Conditional model - density of inputs not modeled
- Consistency: $p(f_1) = \int df_2 \, p(f_1, f_2)$

Covariances

Where does the covariance matrix $\mathbf{K}$ come from?

- Covariance matrix constructed from *covariance function*:

$$\mathbf{K}_{ij} = K(x_i, x_j)$$

- Covariance function characterizes correlations between different points in the process:

$$K(x, x') = \mathcal{E}[f(x)f(x')]$$

- Must produce positive semidefinite covariance matrices $\mathbf{v}^\top \mathbf{K} \mathbf{v} \geq 0$

- Ensures consistency

Squared exponential (SE) covariance

$$K(x, x') = {\sigma_0}^2 \exp\left[-\frac{1}{2}\left(\frac{x - x'}{\lambda}\right)^2\right]$$

- Intuition: function variables close in input space are highly correlated, whilst those far away are uncorrelated
- λ, σ_0 — hyperparameters. λ: lengthscale, σ_0: amplitude
- Stationary: $K(x, x') = K(x - x')$ — invariant to translations
- Very smooth sample functions — infinitely differentiable

Matérn class of covariances

$$K(x,x') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}|x-x'|}{\lambda}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}|x-x'|}{\lambda}\right)$$

where K_ν is a modified Bessel function.

- Stationary, isotropic
- $\nu \to \infty$: SE covariance
- Finite ν: much rougher sample functions
- $\nu = 1/2$: $K(x,x') = \exp(-|x-x'|/\lambda)$, OU process, very rough sample functions

Nonstationary covariances

- Linear covariance: $K(x,x') = \sigma_0^2 + xx'$
- Brownian motion (Wiener process): $K(x,x') = \min(x,x')$
- Periodic covariance: $K(x,x') = \exp\left(-\frac{2\sin^2\left(\frac{x-x'}{2}\right)}{\lambda^2}\right)$
- Neural network covariance

Constructing new covariances from old

There are several ways to combine covariances:

- Sum: $K(x,x') = K_1(x,x') + K_2(x,x')$
  addition of independent processes

- Product: $K(x,x') = K_1(x,x')K_2(x,x')$
  product of independent processes

- Convolution: $K(x,x') = \int dz\, dz'\, h(x,z)K(z,z')h(x',z')$
  blurring of process with kernel h

Prediction with GPs

- We have seen examples of GPs with certain covariance functions
- General properties of covariances controlled by small number of hyperparameters
- Task: prediction from noisy data
- Use GP as a Bayesian prior expressing beliefs about underlying function we are modeling
- Link to data via noise model or likelihood

GP regression with Gaussian noise

Data generated with Gaussian white noise around the function f

$$y = f + \epsilon \qquad \mathcal{E}[\epsilon(x)\epsilon(x')] = \sigma^2\delta(x - x')$$

Equivalently, the noise model, or *likelihood* is:

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma^2\mathbf{I})$$

Integrating over the function variables gives the *marginal likelihood*:

$$\begin{aligned} p(\mathbf{y}) &= \int \mathrm{d}\mathbf{f}\, p(\mathbf{y}|\mathbf{f})p(\mathbf{f}) \\ &= \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I}) \end{aligned}$$

Prediction

N training input and output pairs $(\mathbf{X}, \mathbf{y})$, and T test inputs $\mathbf{X}_T$

Consider joint training and test marginal likelihood:

$$p(\mathbf{y}, \mathbf{y}_T) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{N+T} + \sigma^2 \mathbf{I}) \ , \quad \mathbf{K}_{N+T} = \begin{bmatrix} \mathbf{K}_N & \mathbf{K}_{NT} \\ \mathbf{K}_{TN} & \mathbf{K}_T \end{bmatrix} \ ,$$

Condition on training outputs: $p(\mathbf{y}_T | \mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{\Sigma}_T)$

$$\begin{aligned} \boldsymbol{\mu}_T &= \mathbf{K}_{TN}[\mathbf{K}_N + \sigma^2 \mathbf{I}]^{-1} \mathbf{y} \\ \boldsymbol{\Sigma}_T &= \mathbf{K}_T - \mathbf{K}_{TN}[\mathbf{K}_N + \sigma^2 \mathbf{I}]^{-1} \mathbf{K}_{NT} + \sigma^2 \mathbf{I} \end{aligned}$$

Gives correlated predictions. Defines a predictive Gaussian *process*

Prediction

- Often only marginal variances $(\mathrm{diag}\, \mathbf{\Sigma}_T)$ are required — sufficient to consider a single test input $\mathbf{x}_*$:

$$\mu_* = \mathbf{K}_{*N}[\mathbf{K}_N + \sigma^2 \mathbf{I}]^{-1}\mathbf{y}$$
$$\sigma_*^2 = K_* - \mathbf{K}_{*N}[\mathbf{K}_N + \sigma^2 \mathbf{I}]^{-1}\mathbf{K}_{N*} + \sigma^2 \; .$$

- Mean predictor is a linear predictor: $\mu_* = \mathbf{K}_{*N}\boldsymbol{\alpha}$

- Inversion of $\mathbf{K}_N + \sigma^2 \mathbf{I}$ costs $\mathcal{O}(N^3)$

- Prediction cost per test case is $\mathcal{O}(N)$ for the mean and $\mathcal{O}(N^2)$ for the variance

Determination of hyperparameters

- Advantage of the probabilistic GP framework — ability to choose hyperparameters and covariances directly from the training data
- Other models, e.g. SVMs, splines etc. require cross validation
- GP: minimize negative $\log$ marginal likelihood $\mathcal{L}(\boldsymbol{\theta})$ wrt hyperparameters and noise level $\boldsymbol{\theta}$:

$$\mathcal{L} = -\log p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{2}\log\det\mathbf{C}(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{y}^{\top}\mathbf{C}^{-1}(\boldsymbol{\theta})\mathbf{y} + \frac{N}{2}\log(2\pi)$$

  where $\mathbf{C} = \mathbf{K} + \sigma^2\mathbf{I}$

- Uncertainty in the function variables $\mathbf{f}$ is taken into account

Gradient based optimization

- Minimization of $\mathcal{L}(\boldsymbol{\theta})$ is a non-convex optimization task
- Standard gradient based techniques, such as CG or quasi-Newton
- Gradients:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2} \operatorname{tr} \mathbf{C}^{-1} \frac{\partial \mathbf{C}}{\partial \theta_i} - \frac{1}{2} \mathbf{y}^\top \mathbf{C}^{-1} \frac{\partial \mathbf{C}}{\partial \theta_i} \mathbf{C}^{-1} \mathbf{y}$$

- Local minima, but usually not much of a problem with few hyperparameters
- Use weighted sums of covariances and let ML choose

Automatic relevance determination[1]

The ARD SE covariance function for multi-dimensional inputs:

$$K(\mathbf{x}, \mathbf{x}') = \sigma_0^2 \exp\left[-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_d - x'_d}{\lambda_d}\right)^2\right]$$

- Learn an individual lengthscale hyperparameter λ_d for each input dimension x_d
- λ_d determines the relevancy of input feature d to the regression
- If λ_d very large, then the feature is irrelevant

[1]Neal, 1996. MacKay, 1998.

Relationship to generalized linear regression

- Weighted sum of fixed finite set of M basis functions:

$$f(x) = \sum_{m=1}^{M} w_m \phi_m(x) = \mathbf{w}^\top \boldsymbol{\phi}(x)$$

- Place Gaussian prior on weights: $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_w)$

- Defines GP with finite rank M (degenerate) covariance function:

$$K(x, x') = \mathcal{E}[f(x) f(x')] = \boldsymbol{\phi}^\top(x) \mathbf{\Sigma}_w \boldsymbol{\phi}(x')$$

- Function space vs. weight space interpretations

Relationship to generalized linear regression

- General GP — can specify covariance function directly rather than via set of basis functions
- Mercer's theorem: can always decompose covariance function into eigenfunctions and eigenvalues:

$$K(x,x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\psi_i(x')$$

- If sum finite, back to linear regression. Often sum infinite, and no analytical expressions for eigenfunctions
- Power of kernel methods in general (e.g. GPs, SVMs etc.) — project $x \mapsto \boldsymbol{\psi}(x)$ into high or infinite dimensional feature space and still handle computations tractably

Relationship to neural networks

Neural net with one hidden layer of N_H units:

$$f(\mathbf{x}) = b + \sum_{j=1}^{N_H} v_j h(\mathbf{x}; \mathbf{u}_j)$$

h — bounded hidden layer transfer function
(e.g. $h(\mathbf{x}; \mathbf{u}) = \mathrm{erf}(\mathbf{u}^\top \mathbf{x})$)

- If v's and b zero mean independent, and weights $\mathbf{u}_j$ iid, then CLT implies NN $\rightarrow$ GP as $N_H \rightarrow \infty$ [Neal, 1996]

- NN covariance function depends on transfer function h, but is in general non-stationary

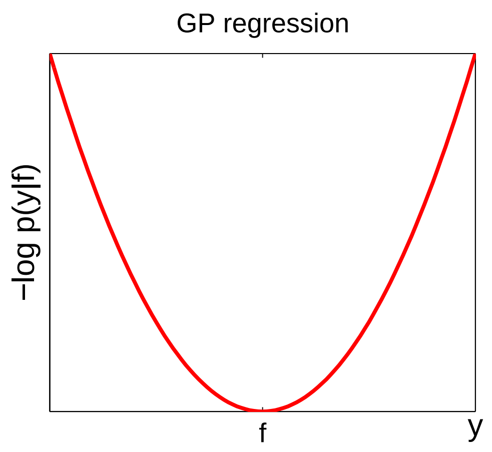
Relationship to spline models

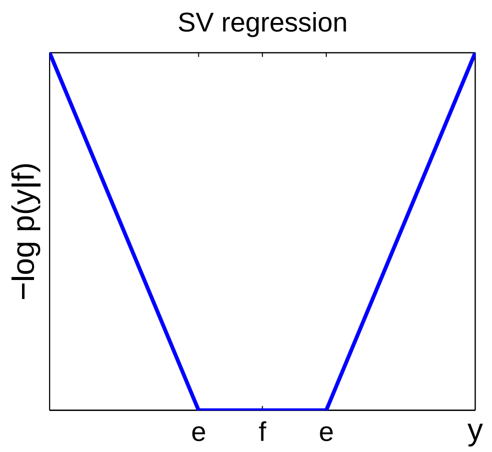
Univariate cubic spline has cost functional:

$$\sum_{n=1}^{N}(f(x_n) - y_n)^2 + \lambda \int_0^1 f''(x)^2 dx$$

- Can give probabilistic GP interpretation by considering RH term as an (improper) GP prior
- Make proper by weak penalty on zeroth and first derivatives
- Can derive a spline covariance function, and full GP machinery can be applied to spline regression (uncertainties, ML hyperparameters)
- Penalties on derivatives — equivalent to specifying the inverse covariance function — no natural marginalization

Relationship to support vector regression





- ϵ-insensitive error function can be considered as a non-Gaussian likelihood or noise model. Integrating over $\mathbf{f}$ becomes intractable

- SV regression can be considered as MAP solution $\mathbf{f}_{\text{MAP}}$ to GP with ϵ-insensitive error likelihood

- Advantages of SVR: naturally sparse solution by QP, robust to outliers. Disadvantages: uncertainties not taken into account, no predictive variances, or learning of hyperparameters by ML

Logistic and probit regression

- Binary classification task: $y = \pm 1$
- GLM likelihood: $p(y = +1|\mathbf{x}, \mathbf{w}) = \pi(\mathbf{x}) = \sigma(\mathbf{x}^\top \mathbf{w})$
- $\sigma(z)$ — sigmoid function such as the logistic or cumulative normal.
- Weight space viewpoint: prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_w)$
- Function space viewpoint: let $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, then likelihood $\pi(\mathbf{x}) = \sigma(f(\mathbf{x}))$, Gaussian prior on $\mathbf{f}$

GP classification

1. Place a GP prior directly on $f(\mathbf{x})$

2. Use a sigmoidal likelihood: $p(y=+1|f) = \sigma(f)$

Just as for SVR, non-Gaussian likelihood makes integrating over $\mathbf{f}$ intractable:

$$p(f_*|\mathbf{y}) = \int \mathrm{d}\mathbf{f}\, p(f_*|\mathbf{f})p(\mathbf{f}|\mathbf{y})$$

where the *posterior* $p(\mathbf{f}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$

Make tractable by using a Gaussian approximation to posterior.
Then prediction:

$$p(y_* = +1|\mathbf{y}) = \int df_*\, \sigma(f_*)p(f_*|\mathbf{y})$$

GP classification

Two common ways to make Gaussian approximation to posterior:

1. Laplace approximation. Second order Taylor approximation about mode of posterior

2. Expectation propagation (EP)[2]. EP can be thought of as approximately minimizing $\mathrm{KL}[p(\mathbf{f}|\mathbf{y})||q(\mathbf{f}|\mathbf{y})]$ by an iterative procedure.

- Kuss and Rasmussen [2005] evaluate both methods experimentally and find EP to be significantly superior

- Classification accuracy on digits data sets comparable to SVMs. Advantages: probabilistic predictions, hyperparameters by ML

[2]Minka, 2001

GP latent variable model (GPLVM)[3]

- Probabilistic model for dimensionality reduction: data is set of high dimensional vectors: $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$

- Model each dimension (k) of $\mathbf{y}$ as an independent GP with unknown low dimensional latent inputs: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$

$$p(\{\mathbf{y}_n\}|\{\mathbf{x}_n\}) \propto \prod_k \exp\left[-\frac{1}{2} w_k^2 \mathbf{Y}_k^\top \mathbf{K}^{-1} \mathbf{Y}_k\right]$$

- Maximize likelihood to find latent projections $\{\mathbf{x}_n\}$ (not a true density model for $\mathbf{y}$ because cannot tractably integrate over $\mathbf{x}$).

- Smooth mapping from $\mathbf{x}$ to $\mathbf{y}$ with uncertainties

[3] Lawrence, 2004

Application: GPLVMs for human pose modeling[4]

- High dimensional data points are feature vectors derived from pose information from mo-cap data.
- Features: joint angles, vertical orientation, velocity and accelerations
- GPLVM used to learn low-dimensional trajectories of e.g. base-ball pitch, basketball jump shot
- GPLVM predictive distribution used to make cost function for finding new poses with constraints

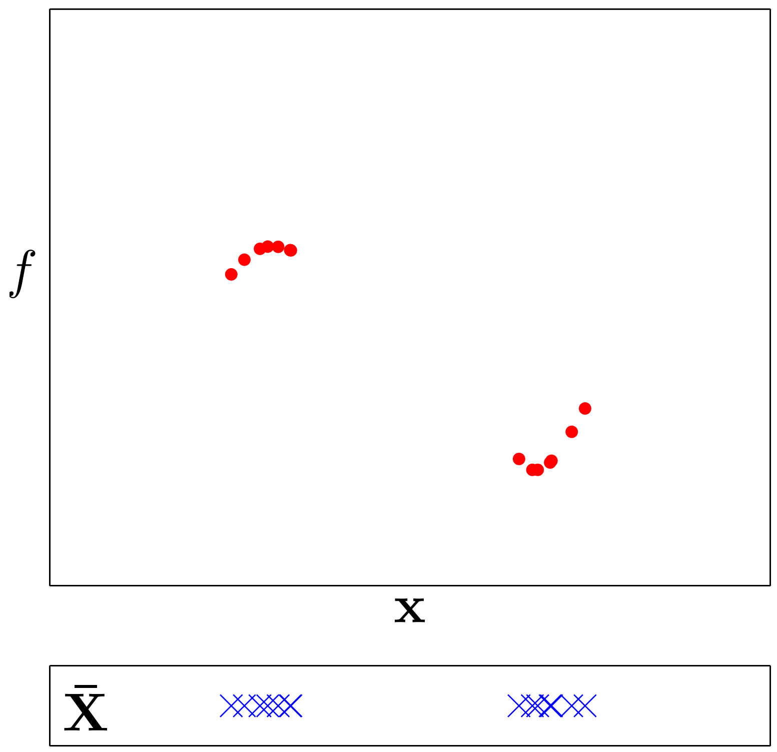
[4]Grochow, Martin, Hertzmann, Popovic, 2004. Style-based inverse kinematics. Demos available from `http://grail.cs.washington.edu/projects/styleik/`

Sparse GP approximations

- Problem for large data sets: training GP $\mathcal{O}(N^3)$, prediction $\mathcal{O}(N^2)$ per test case
- Recent years — many approximations developed — reduce cost to $\mathcal{O}(NM^2)$ training and $\mathcal{O}(M^2)$ prediction per test case
- Based around a low rank (M) covariance approximation
- See Quiñonero Candela and Rasmussen [2005] for a review of regression approximations
- Classification more complicated, so simpler approximations such as IVM[5] may be more suitable

[5]Lawrence et al., 2003

Two stage generative model

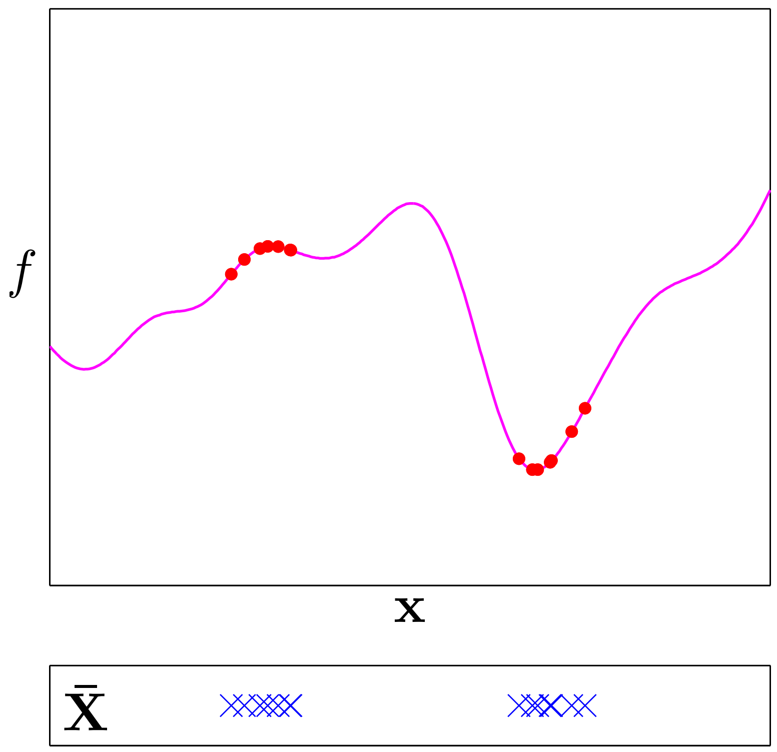


pseudo-input prior

$$p(\bar{\mathbf{f}}|\bar{\mathbf{X}}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_M)$$

1. Choose any set of M (pseudo-) inputs $\bar{\mathbf{X}}$
2. Draw corresponding function values $\bar{\mathbf{f}}$ from prior

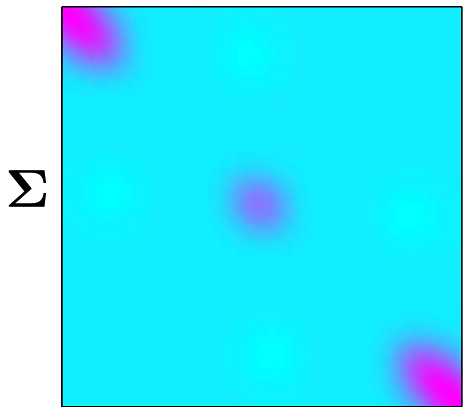
Two stage generative model



conditional

$$p(\mathbf{f}|\bar{\mathbf{f}}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

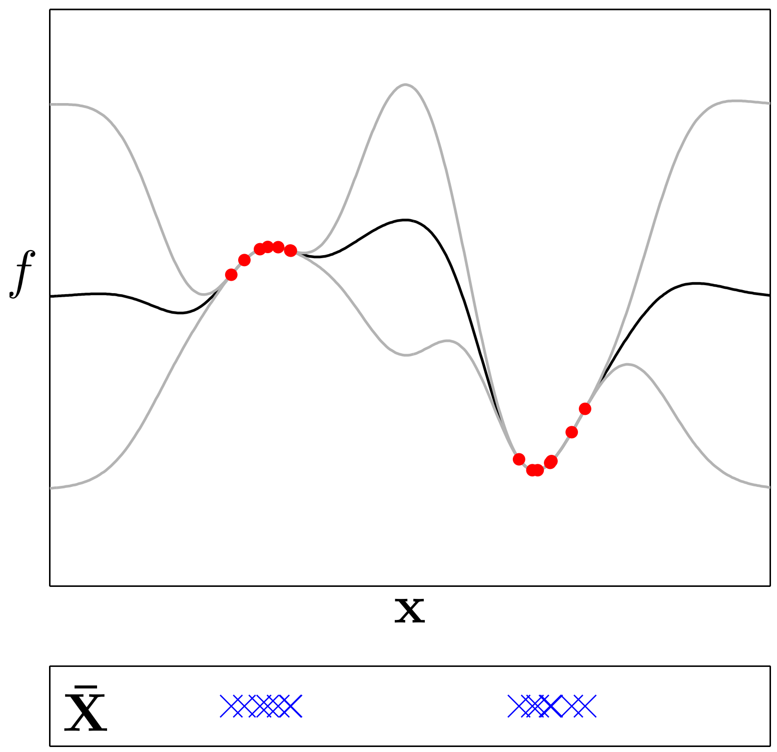
$$\boldsymbol{\mu} = \mathbf{K}_{NM}\mathbf{K}_M^{-1}\bar{\mathbf{f}}$$
$$\boldsymbol{\Sigma} = \mathbf{K}_N - \mathbf{K}_{NM}\mathbf{K}_M^{-1}\mathbf{K}_{MN}$$



3. Draw $\mathbf{f}$ conditioned on $\bar{\mathbf{f}}$

- This two stage procedure defines exactly the same GP prior
- We have not gained anything yet, but it inspires a sparse approximation ...

Factorized approximation

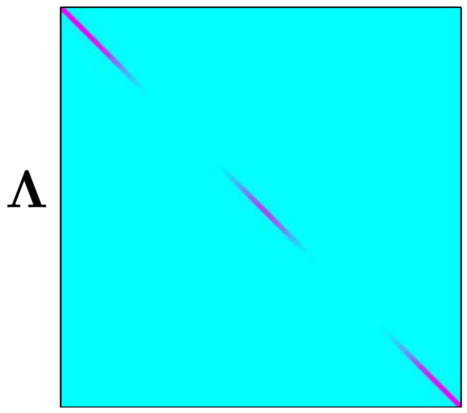


single point conditional

$$p(f_n|\bar{\mathbf{f}}) = \mathcal{N}(\mu_n, \lambda_n)$$

$$\mu_n = \mathbf{K}_{nM}\mathbf{K}_M^{-1}\bar{\mathbf{f}}$$

$$\lambda_n = K_{nn} - \mathbf{K}_{nM}\mathbf{K}_M^{-1}\mathbf{K}_{Mn}$$



Approximate: $p(\mathbf{f}|\bar{\mathbf{f}}) \approx \prod_n p(f_n|\bar{\mathbf{f}}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda})$, $\quad \boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$

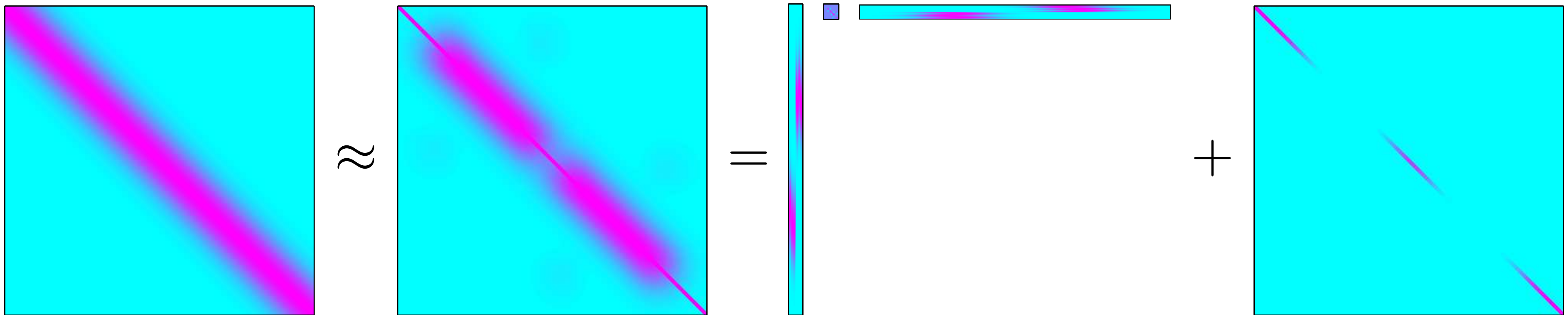
Minimum KL: $\min_{q_n} \mathrm{KL}\left[p(\mathbf{f}|\bar{\mathbf{f}}) \parallel \prod_n q_n(f_n)\right]$

Sparse pseudo-input Gaussian processes (SPGP)[6]

Integrate out $\bar{\mathbf{f}}$ to obtain SPGP prior: $p(\mathbf{f}) = \int \mathrm{d}\bar{\mathbf{f}} \prod_n p(f_n|\bar{\mathbf{f}})\, p(\bar{\mathbf{f}})$

GP prior SPGP/FITC prior

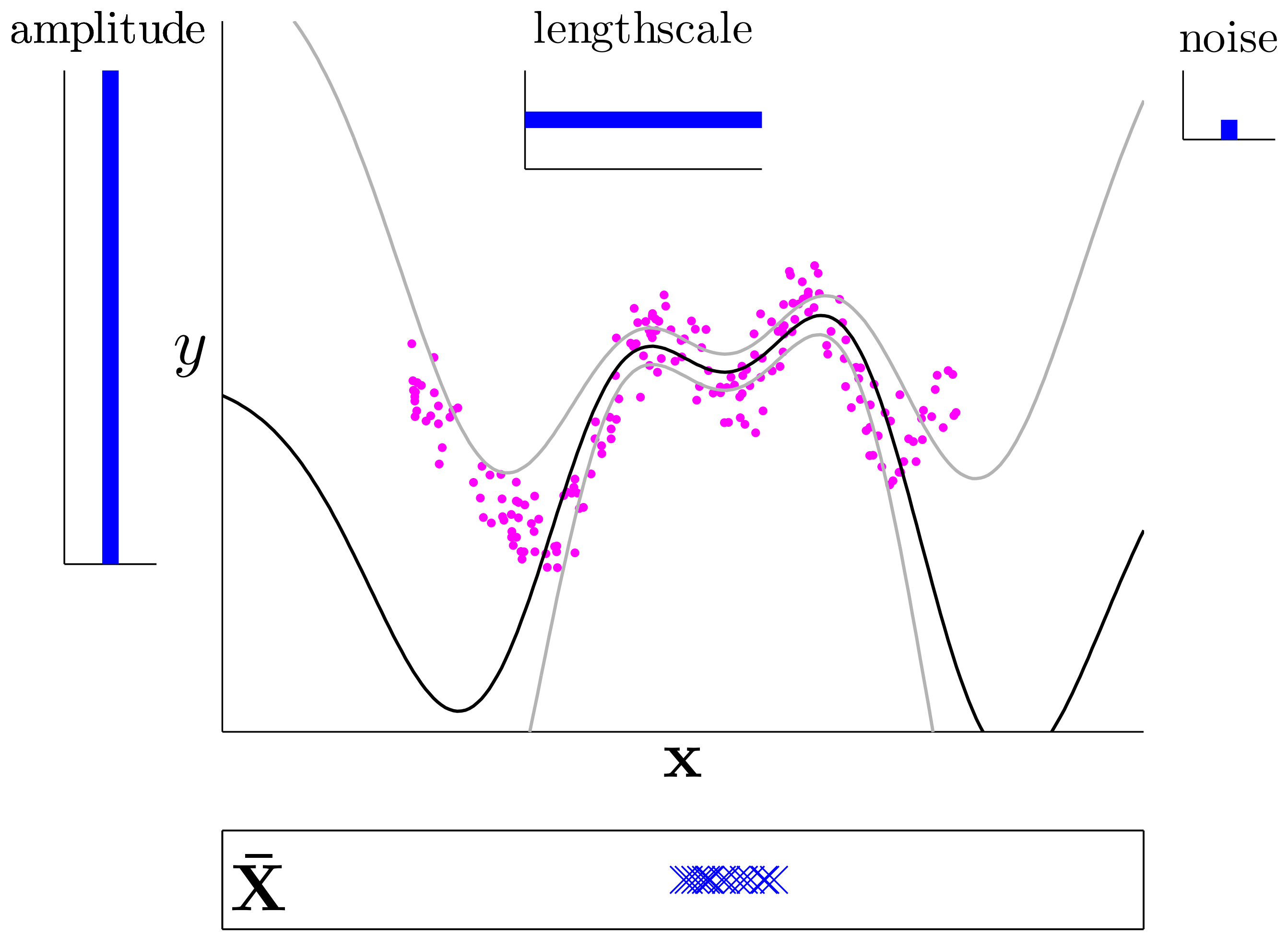
$$\mathcal{N}(\mathbf{0}, \mathbf{K}_N) \quad \approx \quad p(\mathbf{f}) \quad = \mathcal{N}(\mathbf{0}, \mathbf{K}_{NM}\mathbf{K}_M^{-1}\mathbf{K}_{MN} + \boldsymbol{\Lambda})$$



- SPGP/FITC covariance inverted in $\mathcal{O}(M^2N)$. Predictive mean and variance computed in $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ per test case respectively
- SPGP $=$ GP with non-stationary covariance parameterized by $\bar{\mathbf{X}}$

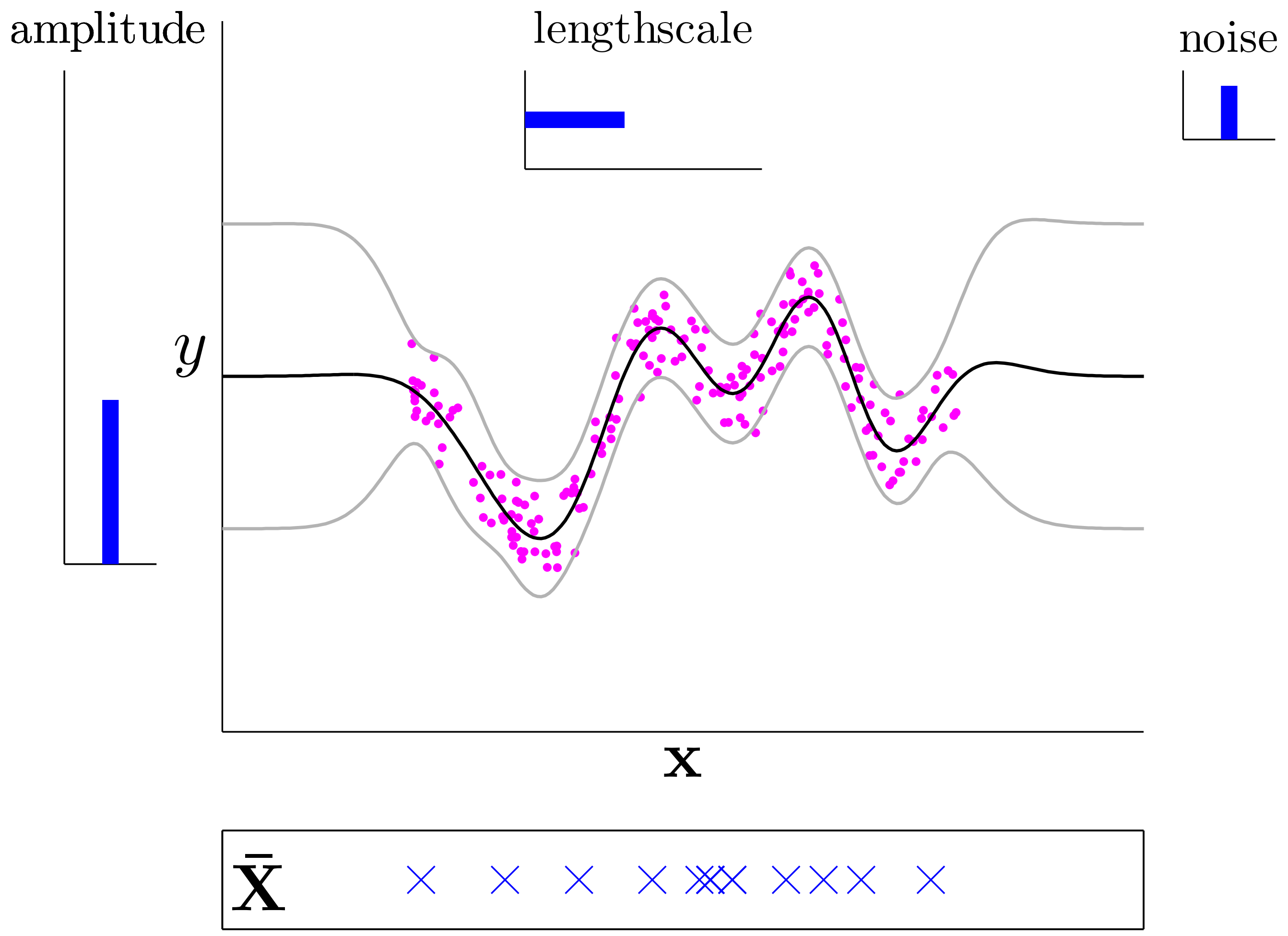
[6]Snelson and Ghahramani, 2005

1D demo



<span style="color:red">Initialize adversarially</span>: amplitude and lengthscale too big
noise too small
pseudo-inputs bunched up

1D demo



Pseudo-inputs and hyperparameters optimized

Future GP directions

- Design of covariance functions to encorporate more specific prior knowledge
- Beyond vectorial input data: structure in the input domain
- Further improvements to sparse GP approximations to scale GPs up for very large data sets
- Beyond regression and classification, e.g. applications of latent variable models such as GPLVM