

Bayesian Supervised Learning and Gaussian Processes

Zoubin Ghahramani

Department of Engineering
University of Cambridge, UK

Machine Learning Department
Carnegie Mellon University, USA

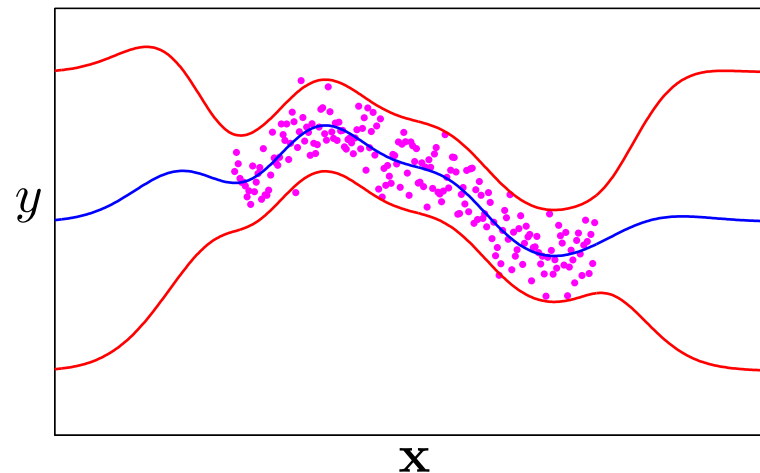
`zoubin@eng.cam.ac.uk`
`http://learning.eng.cam.ac.uk/zoubin/`

IPAM Probabilistic Models of Cognition
Lectures July 2007

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** defines a distribution over functions $p(f)$ which can be used for Bayesian regression:

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

Gaussian Processes

A Gaussian process defines a distribution over functions, $p(f)$, where f is a function mapping some input space \mathcal{X} to \mathbb{R} .

$$f : \mathcal{X} \rightarrow \mathbb{R}.$$

Notice that f can be an infinite-dimensional quantity (e.g. if $\mathcal{X} = \mathbb{R}$)

Let $\mathbf{f} = (f(x_1), \dots, f(x_n))$ be an n -dimensional vector of function values evaluated at n points $x_i \in \mathcal{X}$. Note \mathbf{f} is a random variable.

Definition: $p(f)$ is a **Gaussian process** if for *any* finite subset $\{x_1, \dots, x_n\} \subset \mathcal{X}$, the marginal distribution over that finite subset $p(\mathbf{f})$ has a multivariate Gaussian distribution.

Gaussian process covariance functions

$p(f)$ is a **Gaussian process** if for *any* finite subset $\{x_1, \dots, x_n\} \subset \mathcal{X}$, the marginal distribution over that finite subset $p(\mathbf{f})$ has a multivariate Gaussian distribution.

Gaussian processes (GPs) are parameterized by a **mean function**, $\mu(x)$, and a **covariance function**, $K(x, x')$.

$$p(f(x), f(x')) = \mathbf{N}(\mu, \Sigma)$$

where

$$\mu = \begin{bmatrix} \mu(x) \\ \mu(x') \end{bmatrix} \quad \Sigma = \begin{bmatrix} K(x, x) & K(x, x') \\ K(x', x) & K(x', x') \end{bmatrix}$$

and similarly for $p(f(x_1), \dots, f(x_n))$ where now μ is an $n \times 1$ vector and Σ is an $n \times n$ matrix.

Gaussian process covariance functions

Gaussian processes (GPs) are parameterized by a mean function, $\mu(x)$, and a covariance function, $K(x, x')$.

An example covariance function:

$$K(x_i, x_j) = v_0 \exp \left\{ - \left(\frac{|x_i - x_j|}{r} \right)^\alpha \right\} + v_1 + v_2 \delta_{ij}$$

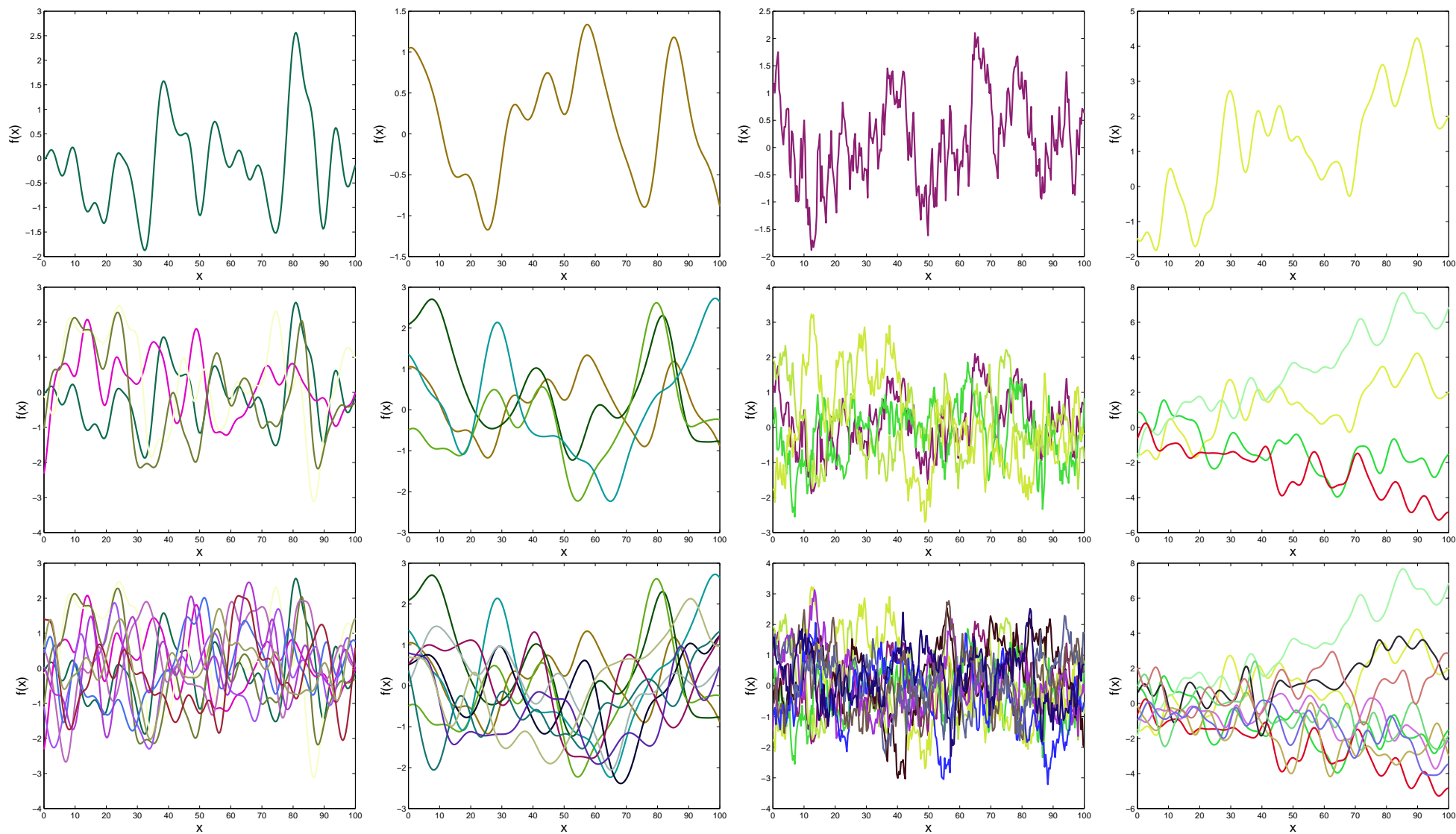
with parameters $(v_0, v_1, v_2, r, \alpha)$

These parameters are interpretable:

v_0	signal variance
v_1	variance of bias
v_2	noise variance
r	lengthscale
α	roughness

Once the mean and covariance functions are defined, everything else about GPs follows from the basic rules of probability applied to multivariate Gaussians.

Samples from GPs with different $K(x, x')$



Using Gaussian processes for nonlinear regression

Imagine observing a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\} = (\mathbf{X}, \mathbf{y})$.

Model:

$$\begin{aligned} y_i &= f(\mathbf{x}_i) + \epsilon_i \\ f &\sim \text{GP}(\cdot|0, K) \\ \epsilon_i &\sim \text{N}(\cdot|0, \sigma^2) \end{aligned}$$

Prior on f is a GP, likelihood is Gaussian, therefore posterior on f is also a GP.

We can use this to make predictions

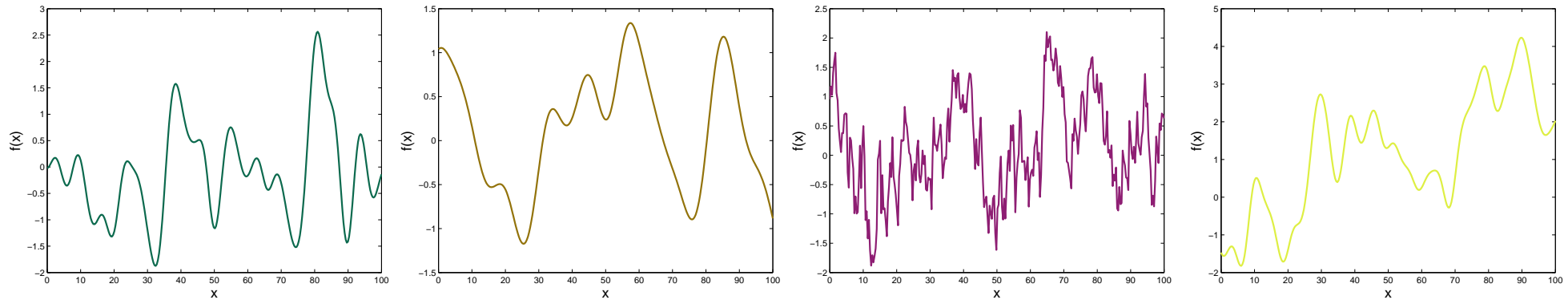
$$p(y_*|\mathbf{x}_*, \mathcal{D}) = \int p(y_*|\mathbf{x}_*, f, \mathcal{D}) p(f|\mathcal{D}) df$$

We can also compute the marginal likelihood (evidence) and use this to compare or tune covariance functions

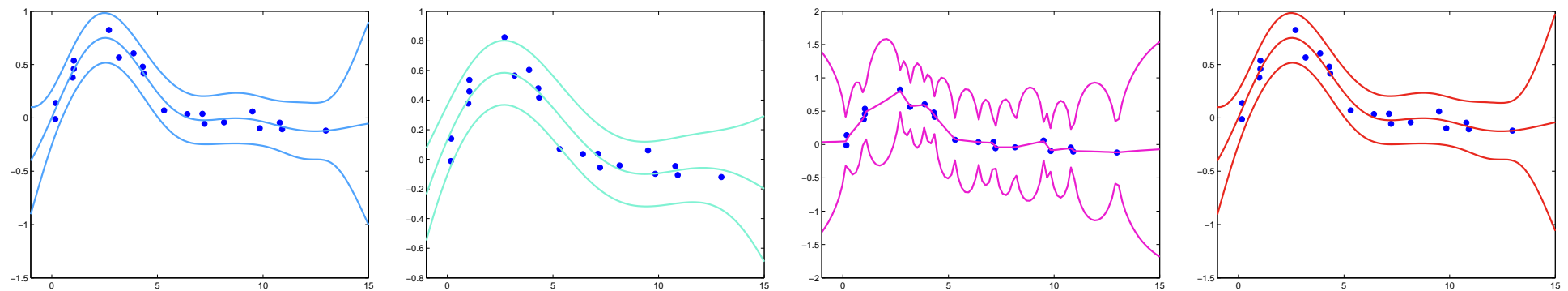
$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|f, \mathbf{X}) p(f) df$$

Prediction using GPs with different $K(x, x')$

A sample from the prior for each covariance function:



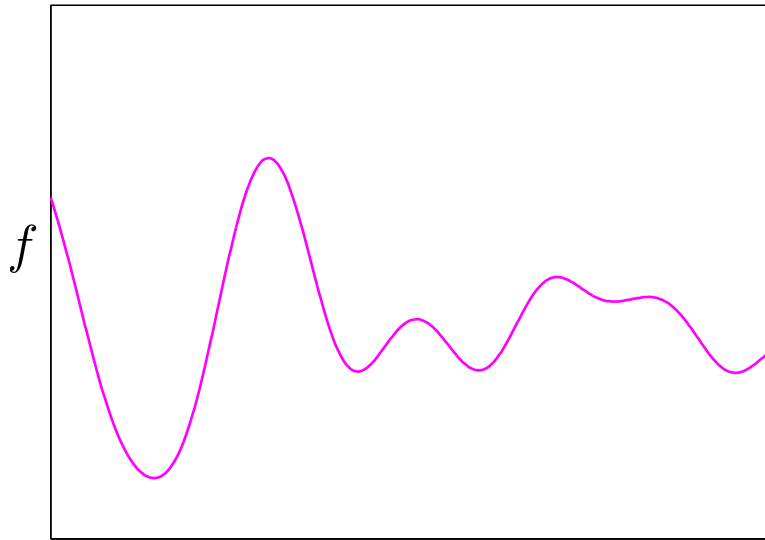
Corresponding predictions, mean with two standard deviations:



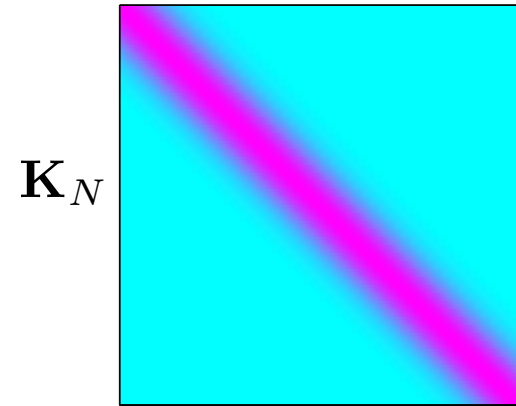
gpdemo

Gaussian process (GP) priors

GP: consistent Gaussian prior on any set of function values $\mathbf{f} = \{f_n\}_{n=1}^N$, given corresponding inputs $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$
one sample function



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

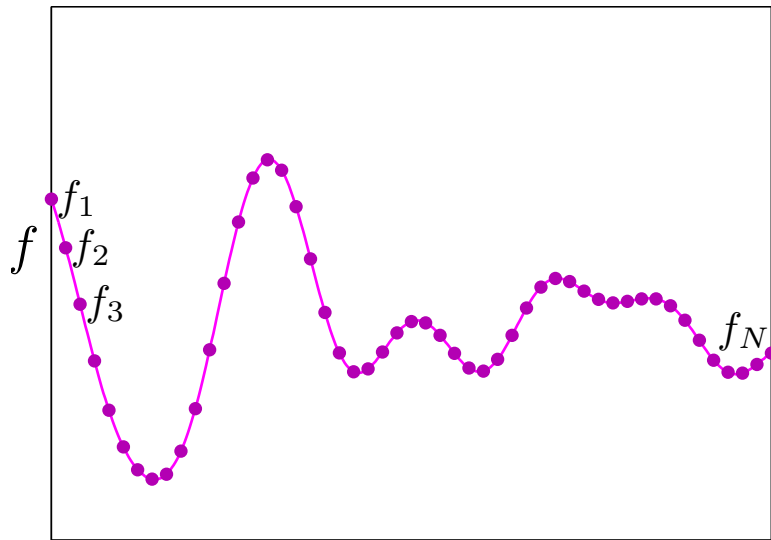


Covariance: $\mathbf{K}_{nn'} = K(\mathbf{x}_n, \mathbf{x}_{n'}; \boldsymbol{\theta})$, hyperparameters $\boldsymbol{\theta}$

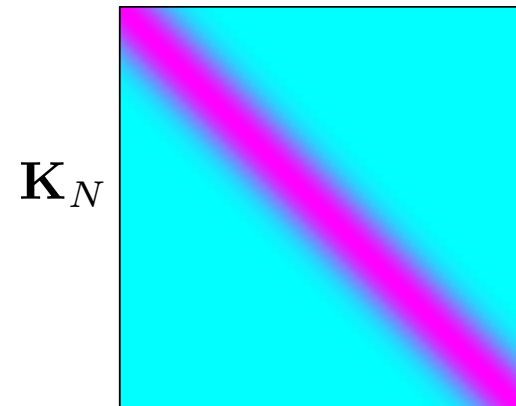
$$\mathbf{K}_{nn'} = v \exp \left[-\frac{1}{2} \sum_{d=1}^D \left(\frac{x_n^{(d)} - x_{n'}^{(d)}}{r_d} \right)^2 \right]$$

Gaussian process (GP) priors

GP: consistent Gaussian prior on any set of function values $\mathbf{f} = \{f_n\}_{n=1}^N$, given corresponding inputs $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$
 N function values



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

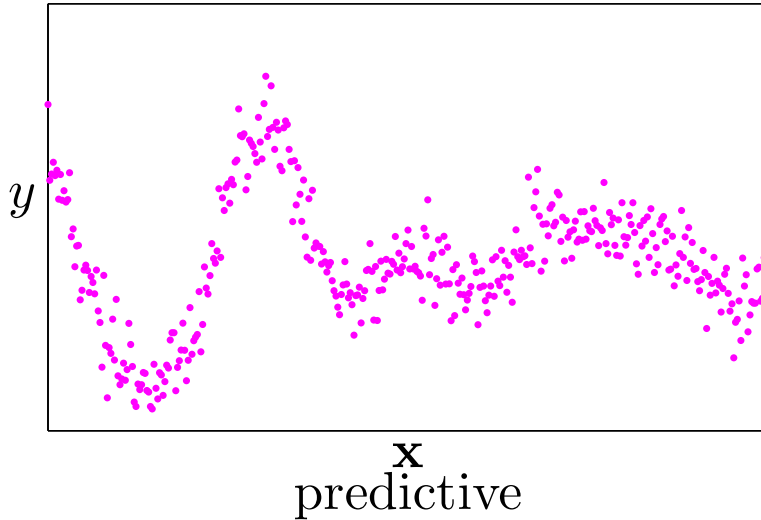


Covariance: $\mathbf{K}_{nn'} = K(\mathbf{x}_n, \mathbf{x}_{n'}; \boldsymbol{\theta})$, hyperparameters $\boldsymbol{\theta}$

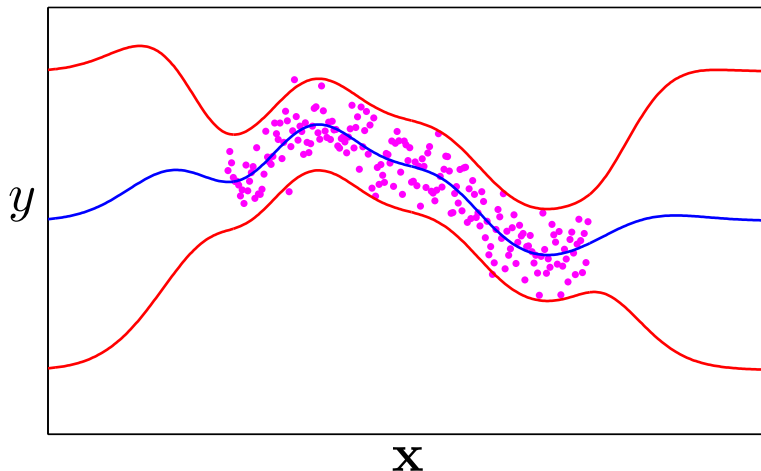
$$\mathbf{K}_{nn'} = v \exp \left[-\frac{1}{2} \sum_{d=1}^D \left(\frac{x_n^{(d)} - x_{n'}^{(d)}}{r_d} \right)^2 \right]$$

GP regression

Gaussian observation noise: $y_n = f_n + \epsilon_n$, where $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$
sample data



marginal likelihood
 $p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_N + \sigma^2 \mathbf{I})$



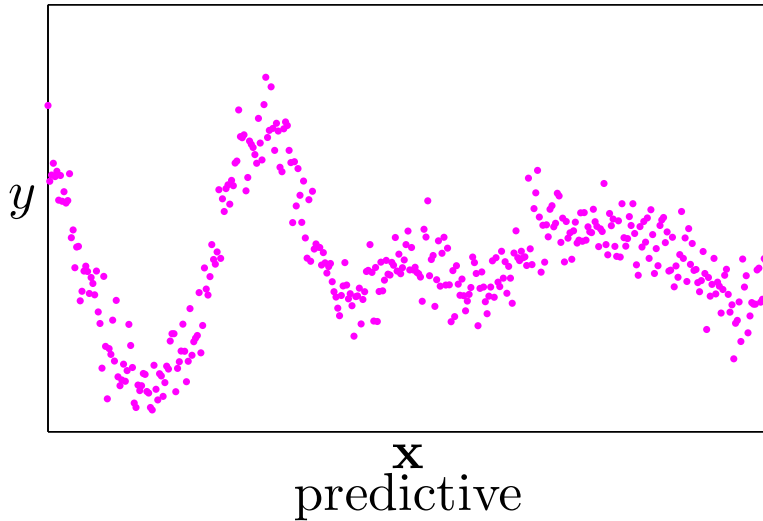
predictive distribution
 $p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\mu_*, \sigma_*^2)$

$$\mu_* = \mathbf{K}_{*N}(\mathbf{K}_N + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$
$$\sigma_*^2 = K_{**} - \mathbf{K}_{*N}(\mathbf{K}_N + \sigma^2 \mathbf{I})^{-1} \mathbf{K}_{N*} + \sigma^2$$

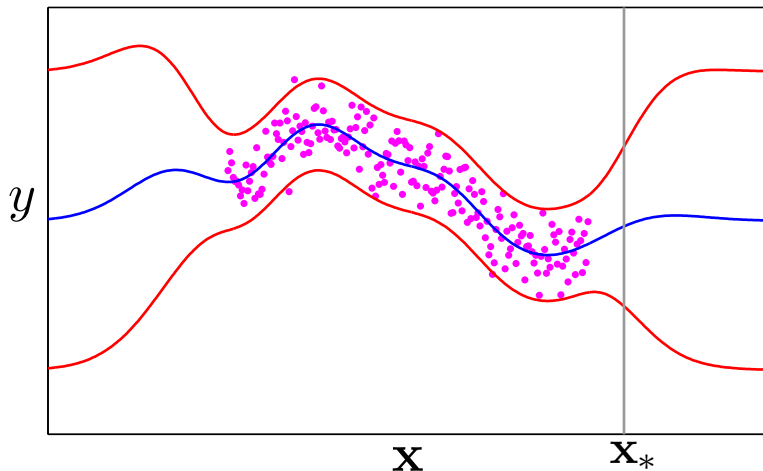
Problem: N^3 computation

GP regression

Gaussian observation noise: $y_n = f_n + \epsilon_n$, where $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$
sample data



marginal likelihood
 $p(\mathbf{y}|\mathbf{X}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_N + \sigma^2 \mathbf{I})$



predictive distribution
 $p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\mu_*, \sigma_*^2)$

$$\mu_* = \mathbf{K}_{*N}(\mathbf{K}_N + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$
$$\sigma_*^2 = K_{**} - \mathbf{K}_{*N}(\mathbf{K}_N + \sigma^2 \mathbf{I})^{-1} \mathbf{K}_{N*} + \sigma^2$$

Problem: N^3 computation

GP learning

Consider the **covariance function** K with hyperparameters $\boldsymbol{\theta} = (v_0, v_1, r_1, \dots, r_d, \alpha)$:

$$K(\mathbf{x}_i, \mathbf{x}_j) = v_0 \exp \left\{ - \sum_{d=1}^D \left(\frac{|x_i^{(d)} - x_j^{(d)}|}{r_d} \right)^\alpha \right\} + v_1$$

Given a data set $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, how do we learn $\boldsymbol{\theta}$?

The **marginal likelihood** is a function of $\boldsymbol{\theta}$

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_N + \sigma^2 \mathbf{I})$$

where its log is:

$$\ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2} \ln \det(\mathbf{K}_N + \sigma^2 \mathbf{I}) - \frac{1}{2} \mathbf{y}^\top (\mathbf{K}_N + \sigma^2 \mathbf{I})^{-1} \mathbf{y} + \text{const}$$

which can be optimized as a function of θ .

Alternatively, one can infer θ using Bayesian methods, which is more costly but immune to overfitting.

From linear regression to GPs:

- **Linear regression** with inputs x_i and outputs y_i : $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$
- Linear regression with M **basis functions**: $y_i = \sum_{m=1}^M \beta_m \phi_m(x_i) + \epsilon_i$
- **Bayesian** linear regression with basis functions:

$$\beta_m \sim \mathcal{N}(\cdot | 0, \lambda_m) \quad (\text{independent of } \beta_\ell, \forall \ell \neq m), \quad \epsilon_i \sim \mathcal{N}(\cdot | 0, \sigma^2)$$

- **Integrating out** the coefficients, β_j , we find:

$$E[y_i] = 0, \quad \text{Cov}(y_i, y_j) = K_{ij} \stackrel{\text{def}}{=} \sum_m \lambda_m \phi_m(x_i) \phi_m(x_j) + \delta_{ij} \sigma^2$$

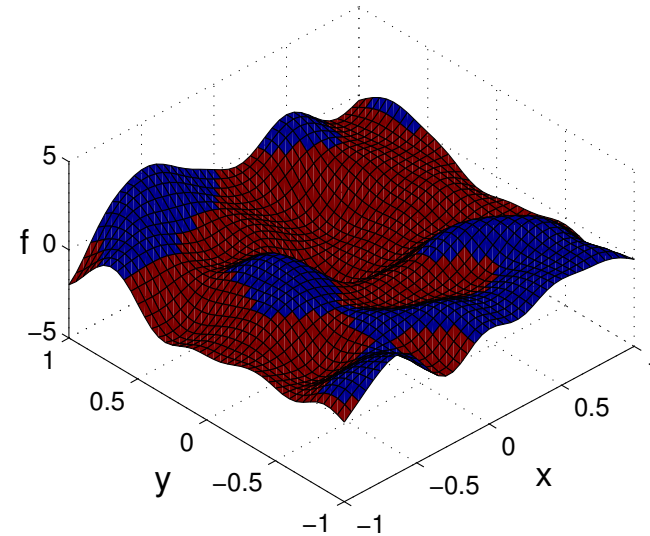
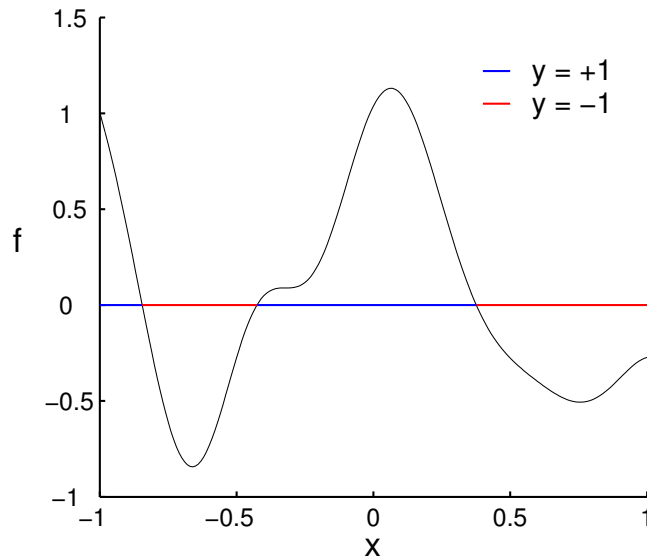
This is a Gaussian process with covariance function $K(x_i, x_j) = K_{ij}$.

This Gaussian process has a finite number (M) of basis functions. Many useful GP covariance functions correspond to infinitely many basis functions.

A multilayer perceptron (neural network) with infinitely many hidden units and Gaussian priors on the weights \rightarrow a GP (Neal, 1992)

Using Gaussian Processes for Classification

Binary classification problem: Given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, with binary class labels $y_i \in \{-1, +1\}$, infer class label probabilities at new points.



There are many ways to relate function values $f(\mathbf{x}_i)$ to class probabilities:

$$p(y|f) = \begin{cases} \frac{1}{1 + \exp(-yf)} \\ \Phi(yf) \\ \mathbf{H}(yf) \\ \epsilon + (1 - 2\epsilon)\mathbf{H}(yf) \end{cases}$$

sigmoid (logistic)
cumulative normal (probit)
threshold
robust threshold

Again: non-Gaussian likelihood function, so we need to use approximate inference methods (Laplace, EP, MCMC).

Sparse Approximations: Speeding up GP learning

(Snelson and Ghahramani, 2005, 2006, 2007)

We can approximate GP through $M < N$ inducing points $\bar{\mathbf{f}}$ to obtain this Sparse Pseudo-input Gaussian process (SPGP) prior: $p(\mathbf{f}) = \int d\bar{\mathbf{f}} \prod_n p(f_n|\bar{\mathbf{f}}) p(\bar{\mathbf{f}})$

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

- SPGP covariance inverted in $\mathcal{O}(M^2N) \ll \mathcal{O}(N^3) \Rightarrow$ much faster
- SPGP = GP with non-stationary covariance parameterized by $\bar{\mathbf{X}}$
- Given data $\{\mathbf{X}, \mathbf{y}\}$ with noise σ^2 , predictive mean and variance can be computed in $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ per test case respectively

Builds on a large lit on sparse GPs (see Quiñonero Candela and Rasmussen, 2006).

Feature Selection

Example: classification

$$\begin{aligned} \text{input } \mathbf{x} &= (x_1, \dots, x_D) \in \mathbb{R}^D \\ \text{output } y &\in \{+1, -1\} \end{aligned}$$

2^D possible subsets of relevant input features.

One approach, consider all models $m \in \{0, 1\}^D$ and find

$$\hat{m} = \operatorname{argmax}_m p(\mathcal{D}|m)$$

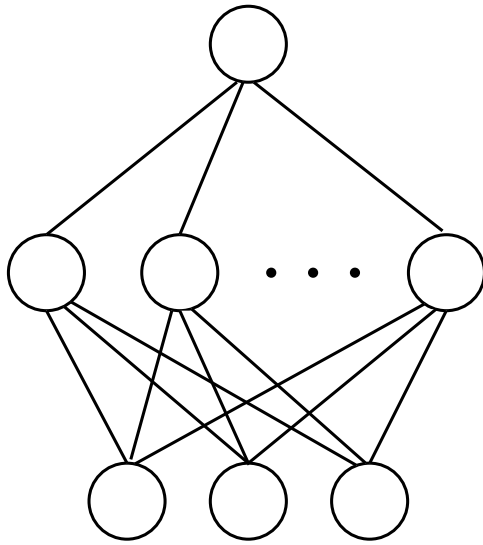
Problems: intractable, overfitting, we should really average

Feature Selection

- Why are we doing feature selection?
- What does it cost us to keep all the features?
- Usual answer (overfitting) does not apply to fully Bayesian methods, since they don't involve any fitting.
- We should only do feature selection if there is a cost associated with measuring features or predicting with many features.

Note: Radford Neal won the NIPS feature selection competition using Bayesian methods that used 100% of the features.

Feature Selection: Automatic Relevance Determination



Bayesian neural network

Data: $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N = (X, \mathbf{y})$

Parameters (weights): $\boldsymbol{\theta} = \{\{w_{ij}\}, \{v_k\}\}$

prior $p(\boldsymbol{\theta}|\boldsymbol{\alpha})$

posterior $p(\boldsymbol{\theta}|\boldsymbol{\alpha}, \mathcal{D}) \propto p(\mathbf{y}|X, \boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\alpha})$

evidence $p(\mathbf{y}|X, \boldsymbol{\alpha}) = \int p(\mathbf{y}|X, \boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\alpha}) d\boldsymbol{\theta}$

prediction $p(y'|\mathcal{D}, \mathbf{x}', \boldsymbol{\alpha}) = \int p(y'|\mathbf{x}', \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\alpha}) d\boldsymbol{\theta}$

Automatic Relevance Determination (ARD):

Let the weights from feature x_d have variance α_d^{-1} : $p(w_{dj}|\alpha_d) = \mathcal{N}(0, \alpha_d^{-1})$

Let's think about this:

$\alpha_d \rightarrow \infty$	variance $\rightarrow 0$	weights $\rightarrow 0$	(irrelevant)
$\alpha_d \ll \infty$	finite variance	weight can vary	(relevant)

ARD: optimize $\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} p(\mathbf{y}|X, \boldsymbol{\alpha})$.

During optimization some α_d will go to ∞ , so the model will discover irrelevant inputs.

Feature Selection using ARD in GPs

Problem: Often there are *many* possible inputs that might be relevant to predicting a particular output. We need algorithms that automatically decide which inputs are relevant.

Automatic Relevance Determination:

Consider this covariance function:

$$\mathbf{K}_{nn'} = v \exp \left[-\frac{1}{2} \sum_{d=1}^D \left(\frac{x_n^{(d)} - x_{n'}^{(d)}}{r_d} \right)^2 \right]$$

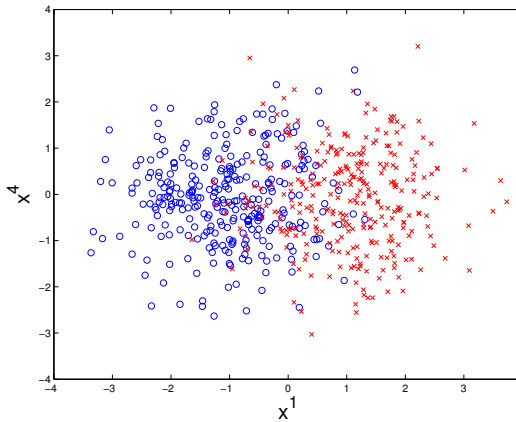
The parameter r_d is the **length scale of the function along input dimension d** .

As $r_d \rightarrow \infty$ the function f varies less and less as a function of $x^{(d)}$, that is, the d th dimension becomes *irrelevant*.

Given data, by learning the lengthscales (r_1, \dots, r_D) it is possible to do automatic feature selection.

An example of ARD for classification

Data set: 6-dimensional data set with three *relevant* features and three *irrelevant* features. For each data point \vec{x}_i , the relevant features depend on its class label: $x_i^1, x_i^2, x_i^3 \sim \mathcal{N}(y_i, 1)$, while the irrelevant features do not: $x_i^4, x_i^5, x_i^6 \sim \mathcal{N}(0, 1)$.



Result: $r_4, r_5, r_6 \rightarrow \infty$ improving the likelihood and classification error rates, compared to a single-lengthscale model.

Methods	single lengthscale	multiple lengthscales
$\log p(\mathbf{y} \mathbf{X}, \boldsymbol{\theta})$	-55.4480	-35.4119
Error rates	0.0600	0.0400

Example from (Kim and Ghahramani, 2004)

More on ARD and feature selection with thousands of inputs: (Qi et al, 2004).

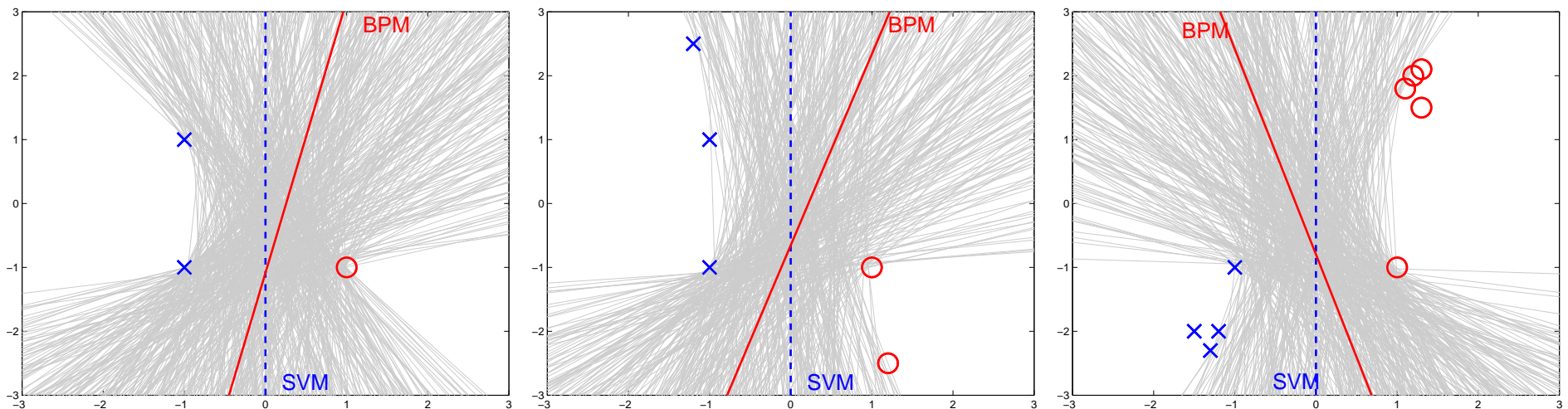
Bayesian Discriminative Modeling

Terminology for classification with inputs \mathbf{x} and classes y :

- **Generative Model:** models prior $p(y)$ and class-conditional density $p(\mathbf{x}|y)$
- **Discriminative Model:** directly models the conditional distribution $p(y|\mathbf{x})$ or the class boundary e.g. $\{\mathbf{x} : p(y = +1|\mathbf{x}) = 0.5\}$

Myth: Bayesian Methods = Generative Models

For example, it is possible to define Bayesian kernel classifiers (e.g. Bayes point machines, and Gaussian processes) analogous to support vector machines (SVMs).



(figure adapted from Minka, 2001)

Appendix

Sparse GP overview

This work contains 2 key ideas:

1. A new sparse Gaussian process approximation based on a small set of M 'pseudo-inputs' ($M \ll N$). This reduces computational complexity to $\mathcal{O}(M^2N)$
2. A gradient based learning procedure for finding the pseudo-inputs and hyperparameters of the Gaussian process, in one joint optimization