A Tutorial on Gaussian Processes
(or why I don’t use SVMs)

Zoubin Ghahramani

Department of Engineering
University of Cambridge, UK

Machine Learning Department
Carnegie Mellon University, USA

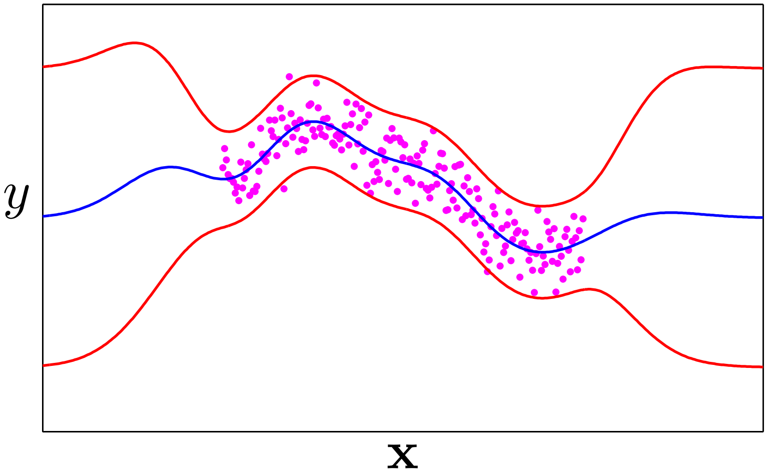
zoubin@eng.cam.ac.uk
http://learning.eng.cam.ac.uk/zoubin/

MLSS 2011

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 $\Re$.

$$f : \mathcal{X} \to \Re.$$

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

Let $\mathbf{f} = (f(x_1), \ldots, f(x_n))$ 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 $\{x_1, \ldots, x_n\} \subset \mathcal{X}$, the marginal distribution over that finite subset $p(\mathbf{f})$ has a multivariate Gaussian distribution.

Gaussian process covariance functions (kernels)

$p(f)$ is a Gaussian process if for *any* finite subset $\{x_1, \ldots, x_n\} \subset \mathcal{X}$, 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, or kernel, $K(x, x')$.

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

where

$$\mu = \left[\begin{array}{c} \mu(x) \\ \mu(x') \end{array} \right] \quad \Sigma = \left[\begin{array}{cc} K(x,x) & K(x,x') \\ K(x',x) & K(x',x') \end{array} \right]$$

and similarly for $p(f(x_1), \ldots, f(x_n))$ 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:

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

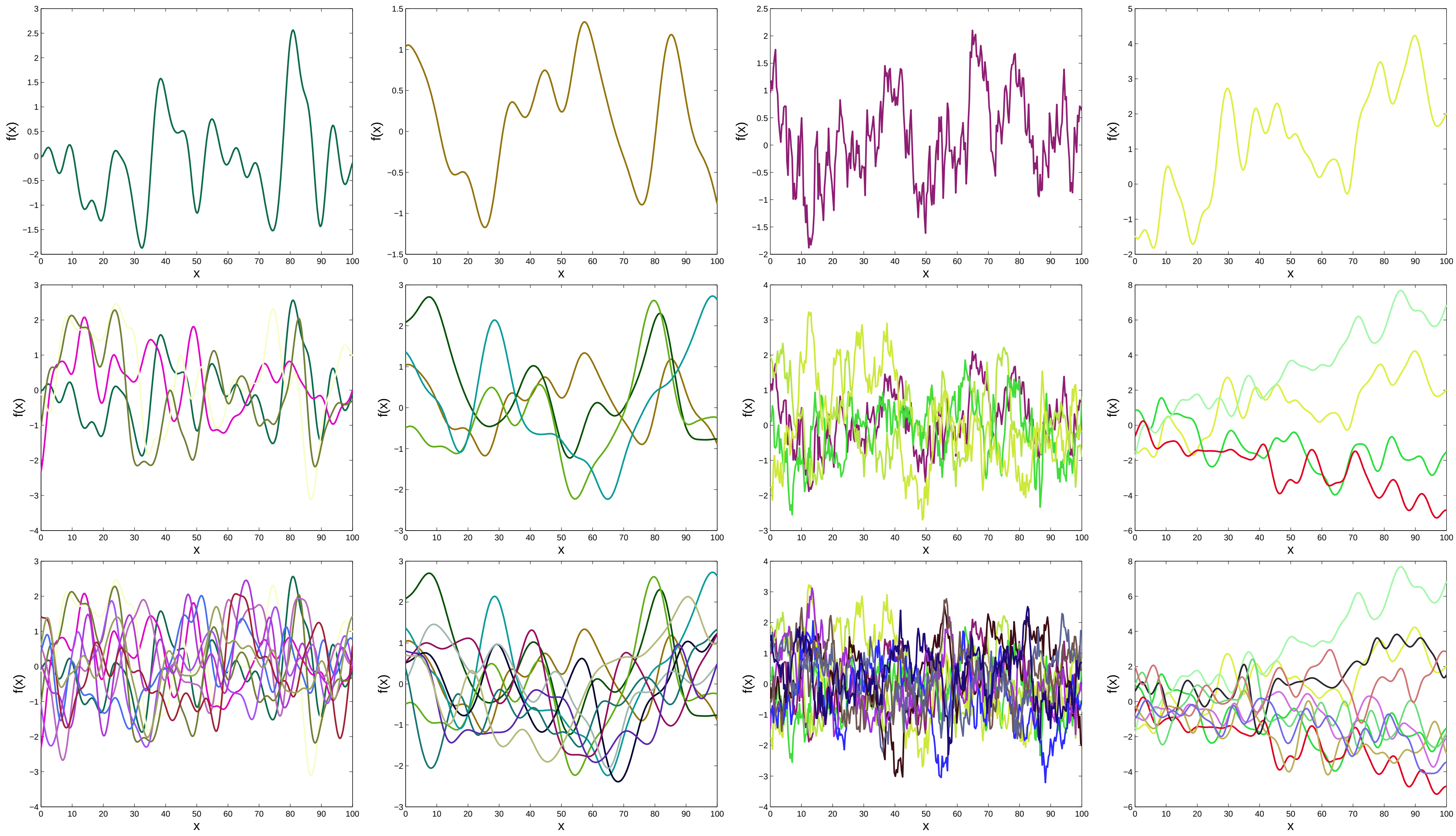
with parameters $(v_0, v_1, v_2, r, \alpha)$

These kernel parameters are interpretable and can be learned from data:

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 mutivariate 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:

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

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

We can use this to make predictions

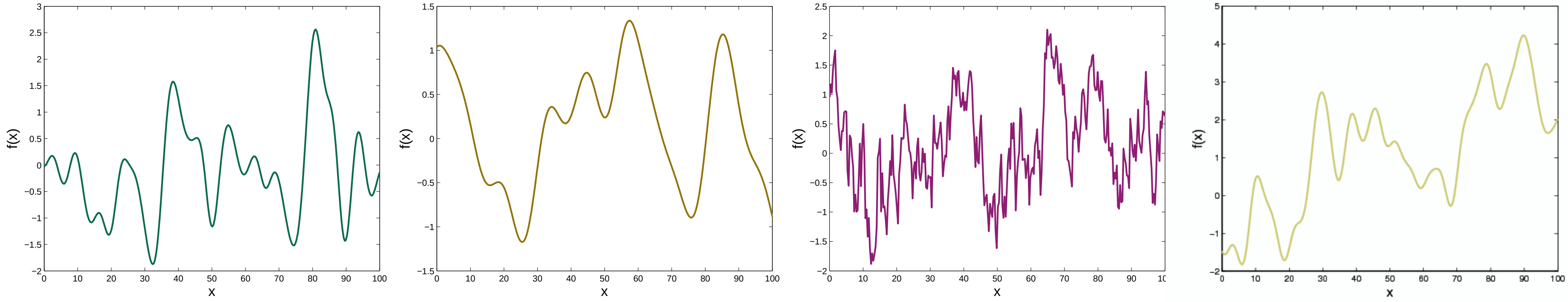
$$p(y_*|\mathbf{x}_*, \mathcal{D}) = \int p(y_*|\mathbf{x}_*, f, \mathcal{D})\, p(f|\mathcal{D})\, df$$

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

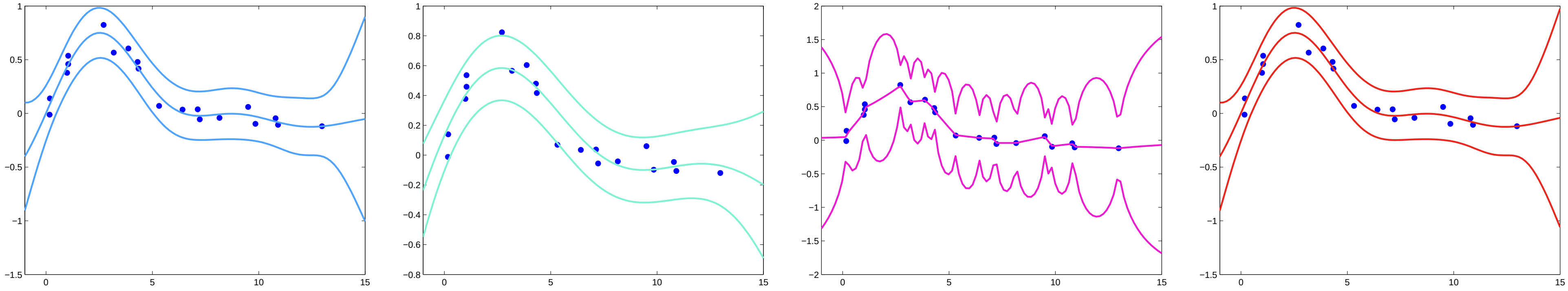
$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|f, \mathbf{X})\, p(f)\, df$$

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

A sample from the prior for each covariance function:



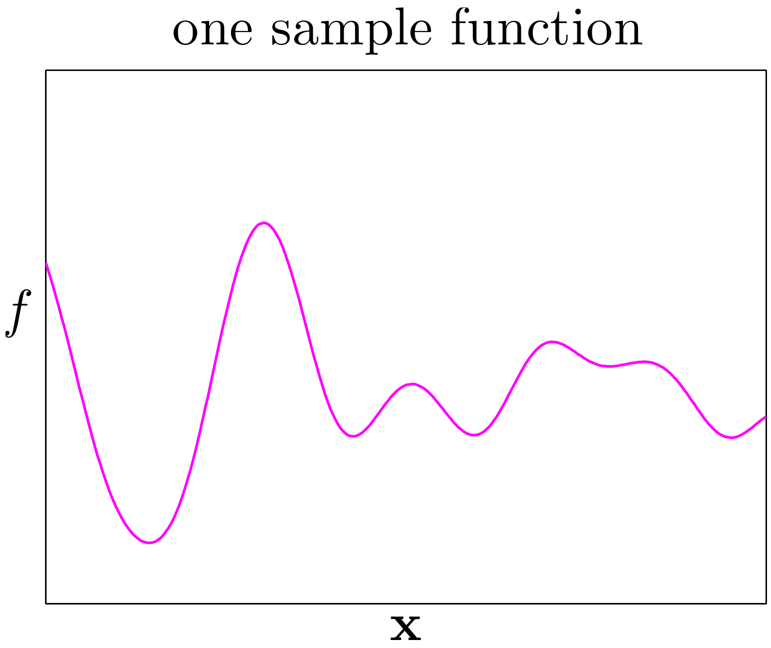
Corresponding predictions, mean with two standard deviations:



`gpdemo`

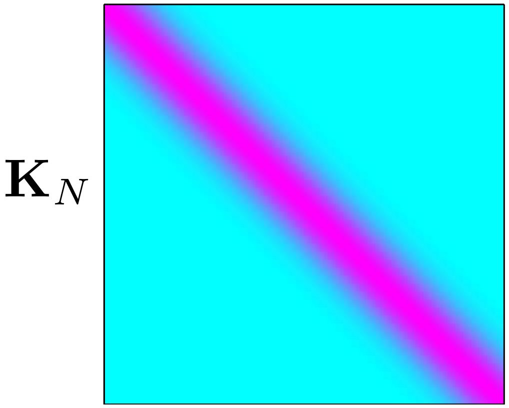
Gaussian process (GP) priors

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



prior

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

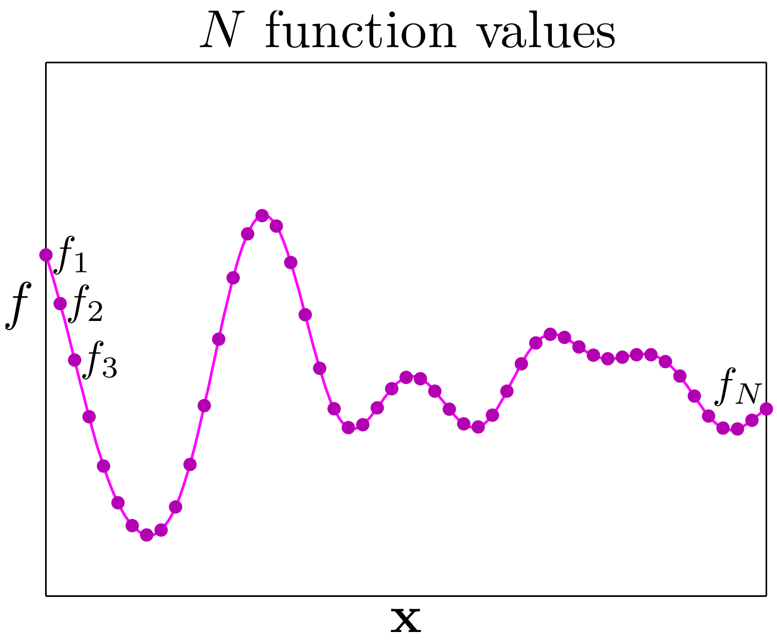


Covariance: $\mathbf{K}_{nn'} = K(\mathbf{x}_n, \mathbf{x}_{n'}\,; \boldsymbol{\theta})$, hyperparameters $\boldsymbol{\theta}$

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

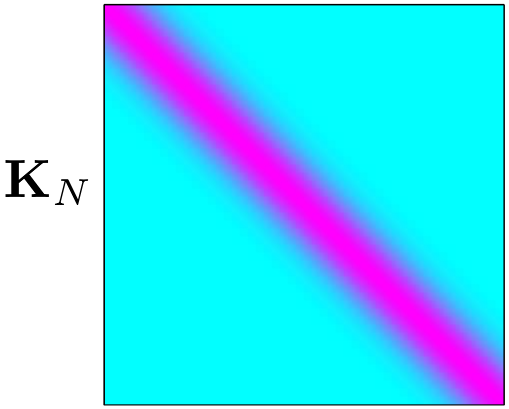
Gaussian process (GP) priors

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



prior

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

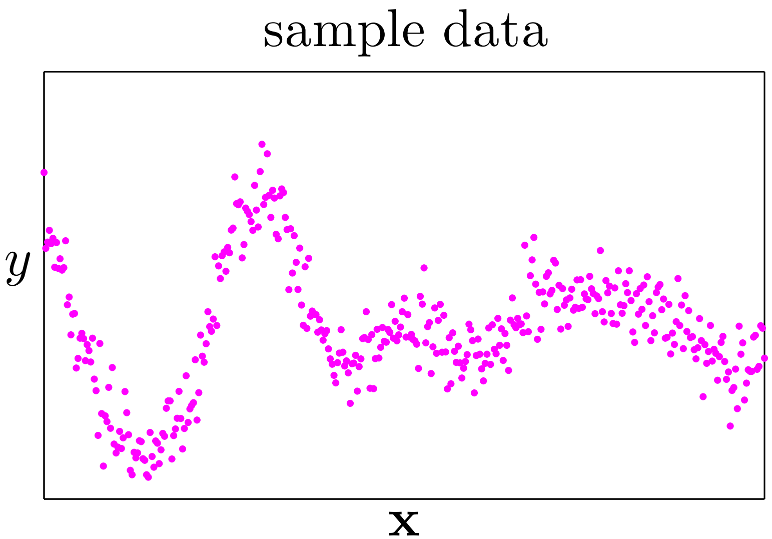


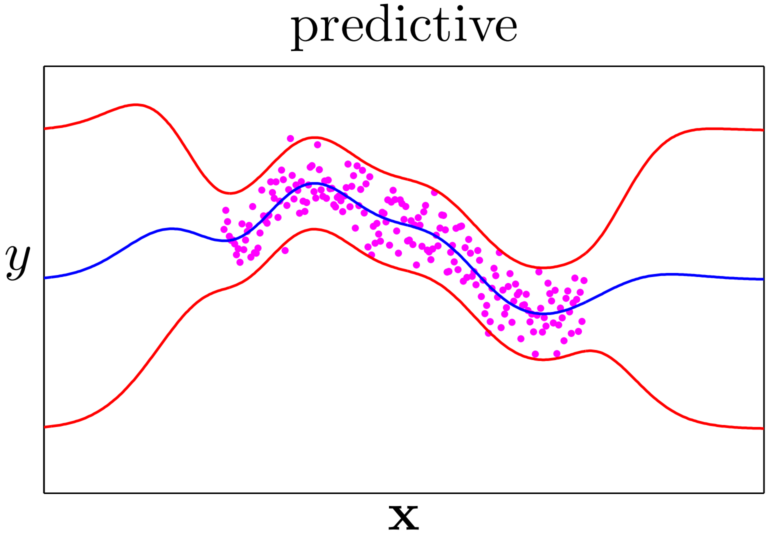
Covariance: $\mathbf{K}_{nn'} = K(\mathbf{x}_n, \mathbf{x}_{n'}\,; \boldsymbol{\theta})$, hyperparameters $\boldsymbol{\theta}$

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

GP regression

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





marginal likelihood

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

predictive distribution

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

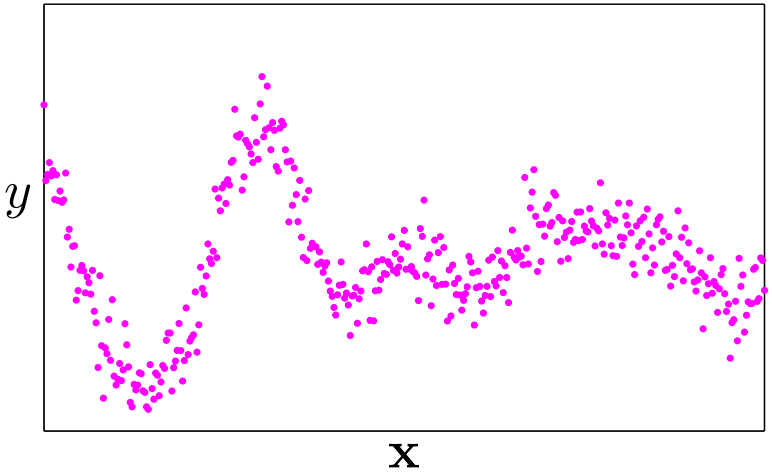
$$\mu_* = \mathbf{K}_{*N}(\mathbf{K}_N + \sigma^2\mathbf{I})^{-1}\mathbf{y}$$

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

GP regression

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

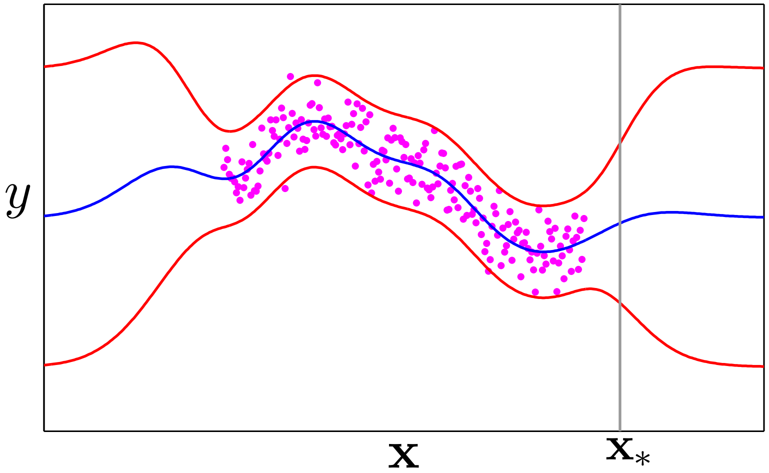
sample data



marginal likelihood

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

predictive



predictive distribution

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

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

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

GP learning the kernel

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

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

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

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

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

where its log is:

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

which can be optimized as a function of $\boldsymbol{\theta}$ and σ.

Alternatively, one can infer $\boldsymbol{\theta}$ 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:

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

- Integrating out the coefficients, β_j, we find:

$$E[y_i] = 0, \qquad Cov(y_i, y_j) = K_{ij} \overset{\text{def}}{=} \sum_{m=1}^{M} \lambda_m \, \phi_m(x_i) \, \phi_m(x_j) + \delta_{ij}\sigma^2$$

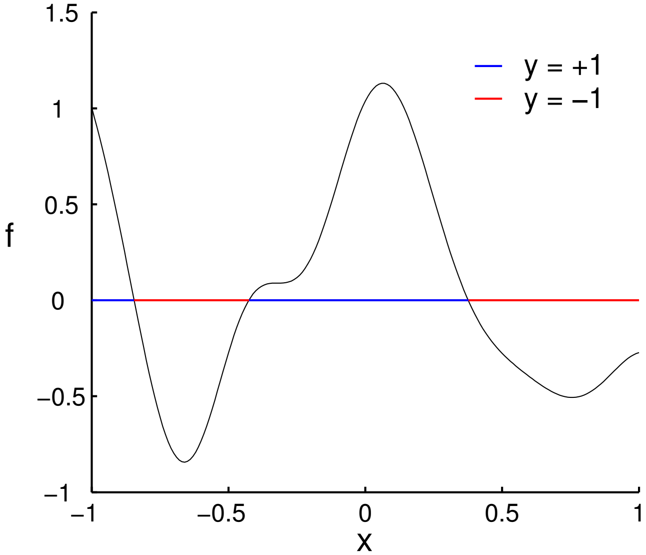
This is a Gaussian process with covariance function $K(x_i, x_j) = K_{ij}$.

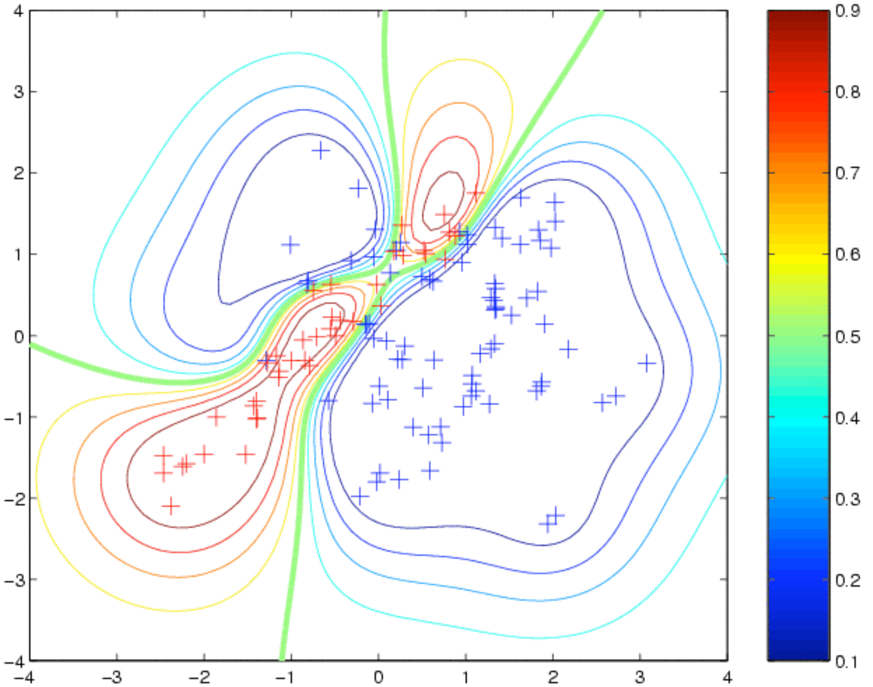
This GP has a finite number (M) of basis functions. Many useful GP kernels correspond to infinitely many basis functions (i.e. infinite-dim feature spaces).

A multilayer perceptron (neural network) with infinitely many hidden units and Gaussian priors on the weights → a GP (Neal, 1996)

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 $f_i = f(\mathbf{x}_i)$ to class probabilities:

$$p(y_i|f_i) = \begin{cases} \frac{1}{1+\exp(-y_i f_i)} & \text{sigmoid (logistic)} \\ \Phi(y_i f_i) & \text{cumulative normal (probit)} \\ \boldsymbol{H}(y_i f_i) & \text{threshold} \\ \epsilon + (1-2\epsilon)\boldsymbol{H}(y_i f_i) & \text{robust threshold} \end{cases}$$

Non-Gaussian likelihood, so we need to use approximate inference methods (Laplace, EP, MCMC).

Support Vector Machines

Consider soft-margin Support Vector Machines:

$$\min_{\mathbf{w}} \; \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i (1 - y_i f_i)_+$$

where $()_+$ is the hinge loss and $f_i = f(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i + w_0$. Let's kernelize this:

$$\mathbf{x}_i \rightarrow \boldsymbol{\phi}(\mathbf{x}_i) = k(\cdot, \mathbf{x}_i), \qquad \mathbf{w} \rightarrow f(\cdot)$$

By reproducing property: $\langle k(\cdot, \mathbf{x}_i), f(\cdot)\rangle = f(\mathbf{x}_i)$.

By representer theorem, solution: $f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}, \mathbf{x}_i)$

Defining $\mathbf{f} = (f_1, \ldots f_N)^T$ note that $\mathbf{f} = \mathbf{K}\boldsymbol{\alpha}$, so $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{f}$

Therefore the regularizer $\frac{1}{2}\|\mathbf{w}\|^2 \rightarrow \frac{1}{2}\|f\|^2_{\mathcal{H}} = \frac{1}{2}\langle f(\cdot), f(\cdot)\rangle_{\mathcal{H}} = \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha} = \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f}$

So we can rewrite the kernelized SVM loss as:

$$\boxed{\min_{\mathbf{f}} \; \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + C\sum_i (1 - y_i f_i)_+}$$

Support Vector Machines and Gaussian Processes

We can write the SVM loss as: $$\min_{\mathbf{f}} \; \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + C\sum_i (1 - y_i f_i)_+$$

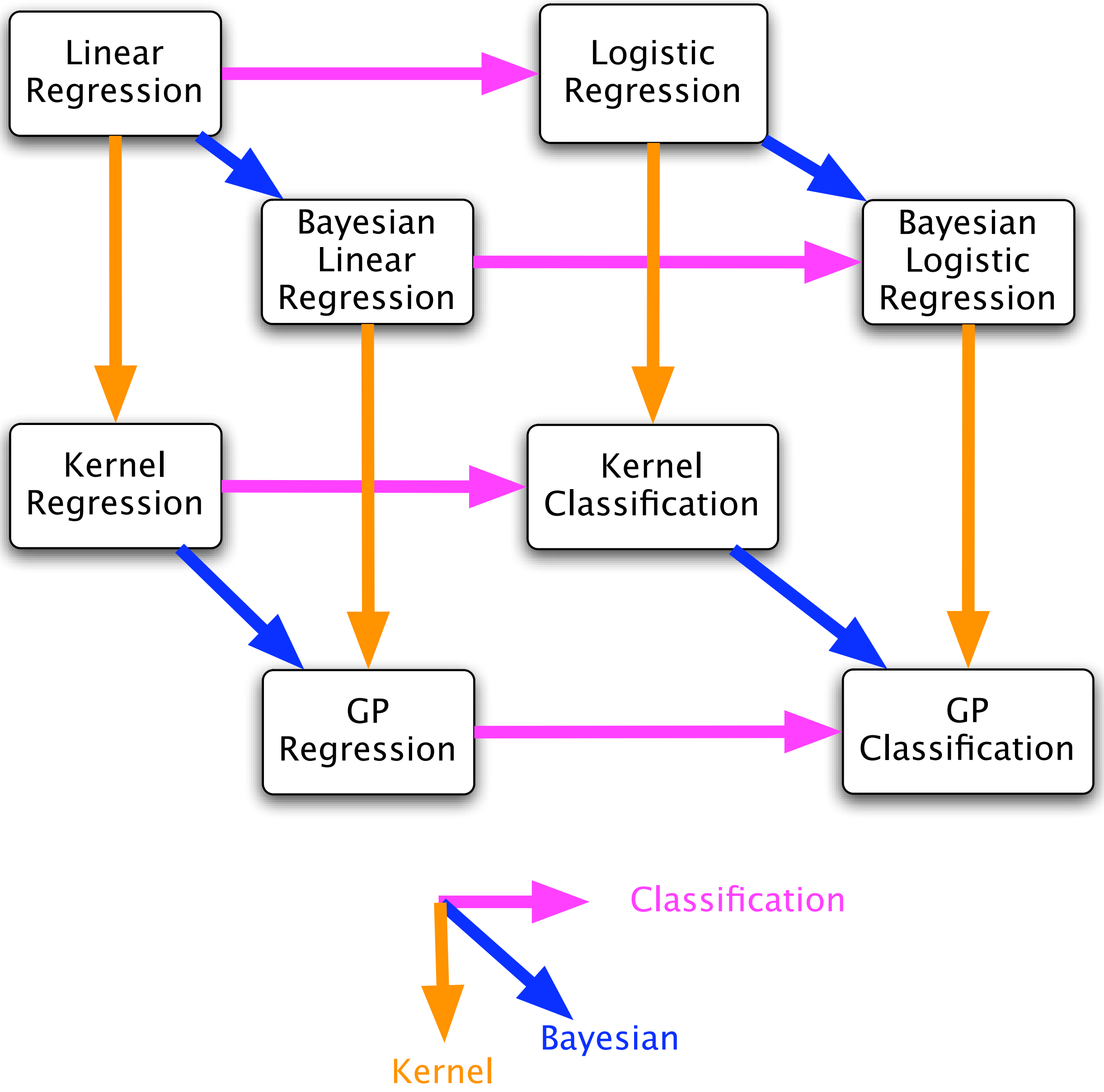
We can write the negative log of a GP likelihood as: $$\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} - \sum_i \ln p(y_i|f_i) + c$$

Equivalent? No.

With Gaussian processes we:

- Handle **uncertainty** in unknown function $\mathbf{f}$ by averaging, not minimization.
- Compute $p(y = +1|\mathbf{x}) \neq p(y = +1|\hat{\mathbf{f}}, \mathbf{x})$.
- Can **learn the kernel parameters** automatically from data, no matter how flexible we wish to make the kernel.
- Can **learn the regularization parameter** C without cross-validation.
- Can incorporate **interpretable** noise models and priors over functions, and can sample from prior to get intuitions about the model assumptions.
- We can combine **automatic feature selection** with learning using ARD.

A picture


Linear Regression
Logistic Regression
Bayesian Linear Regression
Bayesian Logistic Regression
Kernel Regression
Kernel Classification
GP Regression
GP Classification
Classification
Bayesian
Kernel

Matlab Demo: Gaussian Process Classification

```
matlab/gpml-matlab/gpml-demo
```

```
demo_ep_2d
```

```
demo_gpr
```

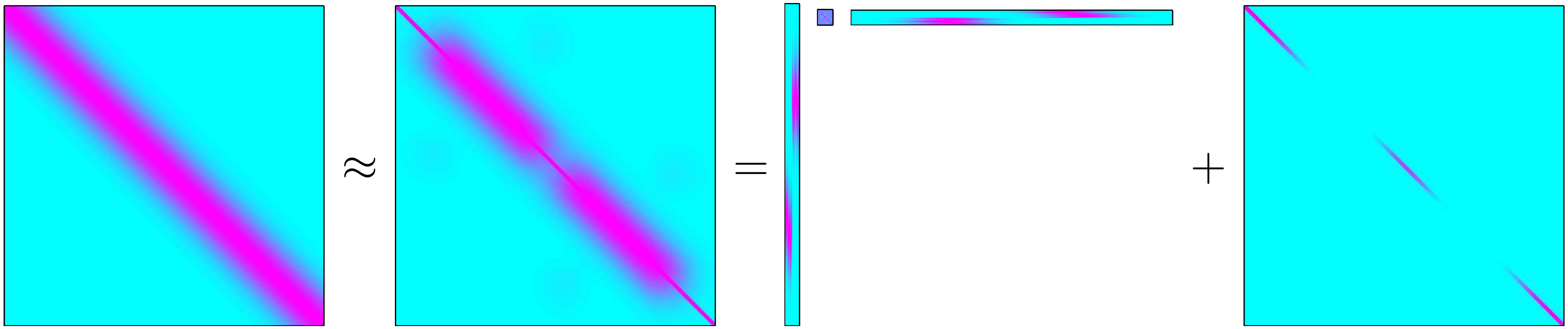
Sparse Approximations: Speeding up GP learning

(Snelson and Ghahramani, 2006a, 2006b; Naish-Guzman and Holden 2008)

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 $\quad$ SPGP prior

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



- 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).

Some Comparisons

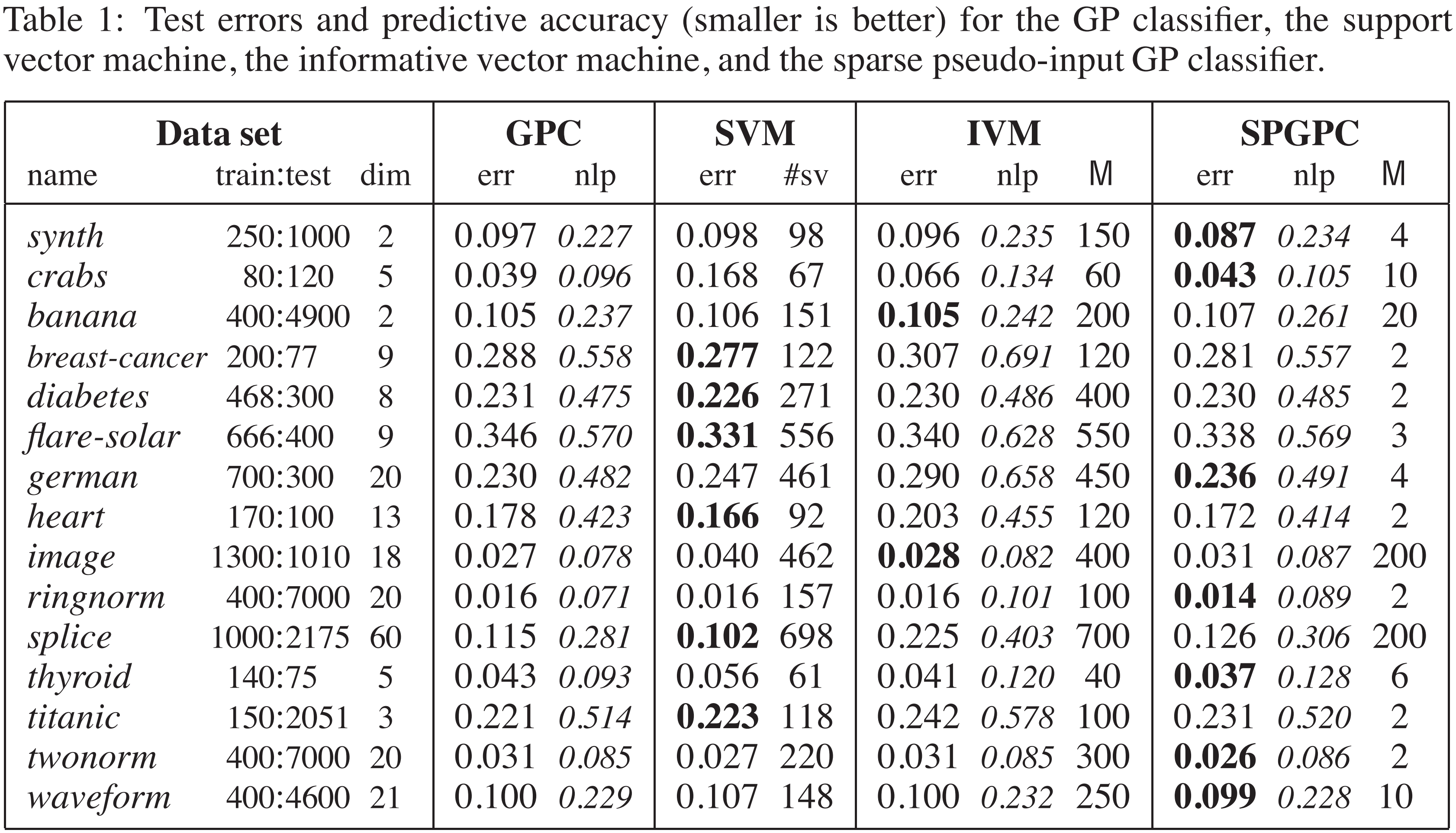


Table 1: Test errors and predictive accuracy (smaller is better) for the GP classifier, the support vector machine, the informative vector machine, and the sparse pseudo-input GP classifier.

Data set			**GPC**		**SVM**		**IVM**			**SPGPC**		
name	train:test	dim	err	nlp	err	#sv	err	nlp	M	err	nlp	M
synth	250:1000	2	0.097	*0.227*	0.098	98	0.096	*0.235*	150	**0.087**	*0.234*	4
crabs	80:120	5	0.039	*0.096*	0.168	67	0.066	*0.134*	60	**0.043**	*0.105*	10
banana	400:4900	2	0.105	*0.237*	0.106	151	**0.105**	*0.242*	200	0.107	*0.261*	20
breast-cancer	200:77	9	0.288	*0.558*	**0.277**	122	0.307	*0.691*	120	0.281	*0.557*	2
diabetes	468:300	8	0.231	*0.475*	**0.226**	271	0.230	*0.486*	400	0.230	*0.485*	2
flare-solar	666:400	9	0.346	*0.570*	**0.331**	556	0.340	*0.628*	550	0.338	*0.569*	3
german	700:300	20	0.230	*0.482*	0.247	461	0.290	*0.658*	450	**0.236**	*0.491*	4
heart	170:100	13	0.178	*0.423*	**0.166**	92	0.203	*0.455*	120	0.172	*0.414*	2
image	1300:1010	18	0.027	*0.078*	0.040	462	**0.028**	*0.082*	400	0.031	*0.087*	200
ringnorm	400:7000	20	0.016	*0.071*	0.016	157	0.016	*0.101*	100	**0.014**	*0.089*	2
splice	1000:2175	60	0.115	*0.281*	**0.102**	698	0.225	*0.403*	700	0.126	*0.306*	200
thyroid	140:75	5	0.043	*0.093*	0.056	61	0.041	*0.120*	40	**0.037**	*0.128*	6
titanic	150:2051	3	0.221	*0.514*	**0.223**	118	0.242	*0.578*	100	0.231	*0.520*	2
twonorm	400:7000	20	0.031	*0.085*	0.027	220	0.031	*0.085*	300	**0.026**	*0.086*	2
waveform	400:4600	21	0.100	*0.229*	0.107	148	0.100	*0.232*	250	**0.099**	*0.228*	10

From (Naish-Guzman and Holden, 2008), using exactly same kernels.

Feature Selection

Example: classification

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

2^D possible subsets of relevant input features.

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

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

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 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 dth dimension becomes *irrelevant*.

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

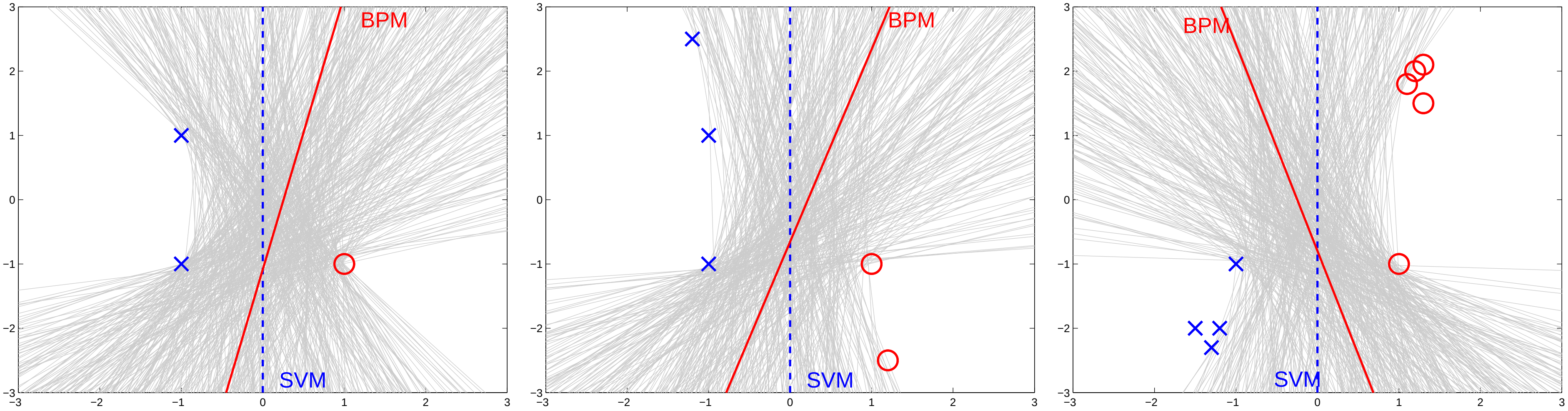
Bayesian Discriminative Modeling

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. $\{\mathbf{x} : p(y = +1|\mathbf{x}) = 0.5\}$

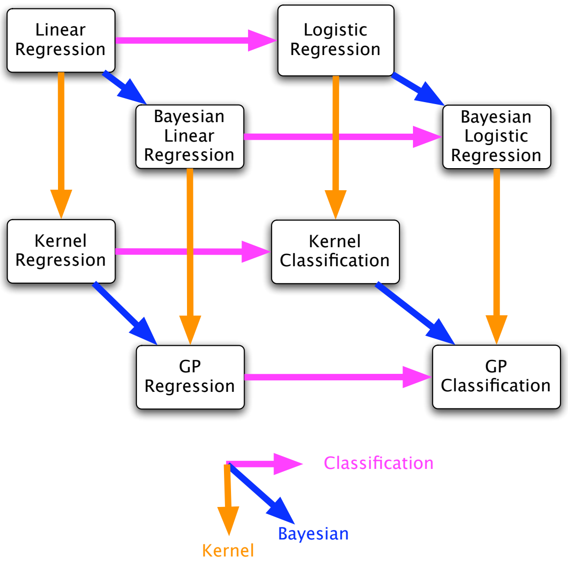
Myth: Bayesian Methods $=$ Generative Models

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



(figure adapted from Minka, 2001)

Conclusions

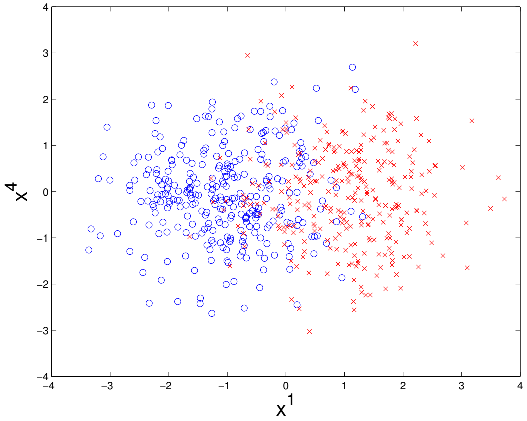


- Gaussian processes define distributions on functions which can be used for nonlinear regression, classification, ranking, preference learning, ordinal regression, etc.
- GPs are closely related to many other models. We can derive them from:
  - Bayesian kernel machines
  - Linear regression with basis functions
  - Infinite multi-layer perceptron neural networks
  - Spline models
- Compared to SVMs, GPs offer several advantages: learning the kernel and regularization parameters, integrated feature selection, fully probabilistic predictions, interpretability.

Appendix

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: $x_i^4, x_i^5, x_i^6 \sim \mathcal{N}(0, 1)$.



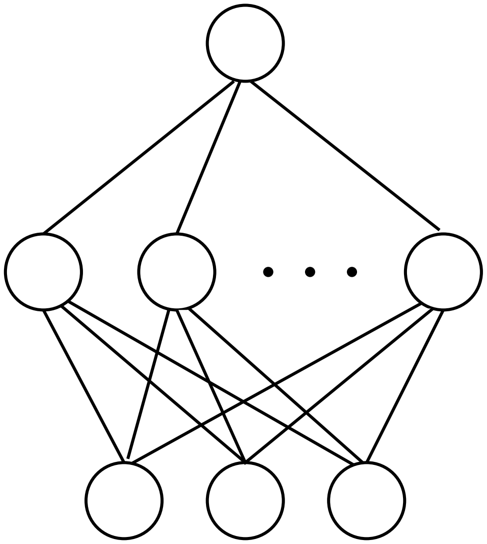
Result: $r_4, r_5, r_6 \rightarrow \infty$ improving the likelihood and classification error rates, compared to a single-lengthscale model.

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).

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}: $\quad p(w_{dj}|\alpha_d) = \mathcal{N}(0, \alpha_d^{-1})$

Let's think about this:	$\alpha_d \to \infty$	variance $\to 0$	weights $\to 0$	(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.

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

References

- Qi, Y., Minka, T.P., Picard, R.W., and Ghahramani, Z. (2004) Predictive Automatic Relevance Determination by Expectation Propagation. In **Twenty-first International Conference on Machine Learning** (ICML-04). Banff, Alberta, Canada.
- Quiñonero-Candela, J. and Rasmussen, C.E. (2005) A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research* **6**:1959.
- Naish-Guzman, A. and Holden, S. (2008) The generalized FITC approximation. Advances in Neural Information Processing Systems 20:1057–1064.
- Neal, R. M. (1996) Bayesian learning for neural networks. Springer Verlag.
- Neal, R. M. (1998). Regression and classification using Gaussian process priors (with discussion). In Bernardo, J. M. et al., editors, **Bayesian statistics 6**, pages 475-501. Oxford University Press.
- O'Hagan, A. (1978). Curve Fitting and Optimal Design for Prediction (with discussion). **Journal of the Royal Statistical Society B**, 40(1):1-42.
- Rasmussen, C.E. and Williams, C.K.I. (2006) Gaussian Processes for Machine Learning. MIT Press.
- Snelson, E. and Ghahramani, Z. (2006a) Sparse Gaussian Processes using Pseudo-Inputs. In **Advances in Neural Information Processing Systems 18** (NIPS-2005).
- Snelson, E. and Ghahramani, Z. (2006b) Variable noise and dimensionality reduction for sparse Gaussian processes. In **Uncertainty in Artifical Intelligence 22** (UAI).

- More information and code at: `http://www.gaussianprocess.org/`