Machine Learning: Waseda University
Kernel Methods

AD

June 2011

Feature space

- Most/all of the algorithms we have discussed rely on a finite dimensional vector of *features* $\Phi(\mathbf{x})$.
- In this way, a model that is linear in $\mathbf{x}$ may be made nonlinear by using a nonlinear mapping $\Phi(\mathbf{x})$.
- In many situations, we only rely on $\Phi(\mathbf{x})$ through the scalar product

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \Phi^{\mathsf{T}}(\mathbf{x}) \Phi\left(\mathbf{x}'\right)$$

- This is a symetric function of its arguments

$$k\left(\mathbf{x}, \mathbf{x}'\right) = k\left(\mathbf{x}', \mathbf{x}\right)$$

Kernels

- A valid kernel is a function $k(\mathbf{x}, \mathbf{x}')$ that corresponds to a scalar (inner) product in some (perhaps infinite dimensional) feature space, i.e. $k(\mathbf{x}, \mathbf{x}') = \Phi^{\mathsf{T}}(\mathbf{x})\,\Phi(\mathbf{x}')$.
- For example assume $\mathbf{x} = (x_1, x_2)$ and

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \left(\mathbf{x}^{\mathsf{T}}\mathbf{x}'\right)^2 \\ &= \left(x_1 x_1' + x_2 x_2'\right)^2 \\ &= x_1^2 \left(x_1'\right)^2 + x_2^2 \left(x_2'\right)^2 + 2 x_1 x_1' x_2 x_2' \\ &= \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2\right) \left(\left(x_2'\right)^2, \sqrt{2} x_1' x_2', \left(x_2'\right)^2\right) \\ &= \Phi^{\mathsf{T}}(\mathbf{x})\,\Phi(\mathbf{x}') \end{aligned}$$

where

$$\Phi(\mathbf{x}) = \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2\right).$$

Positive Semi-definite Kernels

- Losely speaking, a kernel $k(\mathbf{x}, \mathbf{x}')$ can be written as a scalar product possibly in an infinite-dimensional space if it is positive semidefinite; that is for any n, $(\mathbf{x}_1, ..., \mathbf{x}_n) \in \mathcal{X}^n$ and $(\alpha_1, ..., \alpha_n) \in \mathbb{R}^n$ then

$$\sum_i \sum_j \alpha_i \alpha_j k\left(\mathbf{x}_i, \mathbf{x}_i\right) \geq 0$$

- Indeed for continuous symetric positive semidefinite kernel, we have Mercer's theorem. There exists a positive sequence $\{\lambda_i\}$ and functions $\Phi_i\left(\mathbf{x}\right)$ such that

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \sum_{i=1}^{\infty} \lambda_i \Phi_i\left(\mathbf{x}\right) \Phi_i\left(\mathbf{x}'\right).$$

- More later...

Kernel trick

- In many situations, as mentioned earlier, we actually only use $\Phi(\mathbf{x})$ through $\Phi^{\mathsf{T}}(\mathbf{x})\Phi(\mathbf{x}')$.
- Moreover it is often very difficult to design good features $\Phi(\mathbf{x})$.
- Wherever we have $\Phi^{\mathsf{T}}(\mathbf{x})\Phi(\mathbf{x}')$, we can 'kernelize' the algorithm and replace it by $k(\mathbf{x}, \mathbf{x}')$ where $k(\mathbf{x}, \mathbf{x}')$ is a p.s.d. kernel.
- So we can use infinite number of features.
- We can think of $k(\mathbf{x}, \mathbf{x}')$ as a similarity measure: it can be easier to design $k(\mathbf{x}, \mathbf{x}')$ than $\Phi(\mathbf{x})$.

Dual Representation of Linear Regression

- Consider

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^{\mathsf{T}}\Phi(\mathbf{x}_n) - t_n)^2 + \frac{\lambda}{2}\mathbf{w}^{\mathsf{T}}\mathbf{w}$$

  where $\lambda > 0$.

- By setting $\frac{\partial J}{\partial \mathbf{w}} = \mathbf{0}$ we obtain

$$\mathbf{w} = -\frac{1}{\lambda}(\mathbf{w}^{\mathsf{T}}\Phi(\mathbf{x}_n) - t_n)\Phi(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\Phi(\mathbf{x}_n) = \Phi^{\mathsf{T}}\mathbf{a}$$

  where $a_n = -\frac{1}{\lambda}(\mathbf{w}^{\mathsf{T}}\Phi(\mathbf{x}_n) - t_n)$ and Φ is the design matrix

$$\Phi = \begin{pmatrix} \Phi^{\mathsf{T}}(\mathbf{x}_1) \\ \vdots \\ \Phi^{\mathsf{T}}(\mathbf{x}_N) \end{pmatrix}$$

- We now write $\mathbf{w} = \Phi^{\mathsf{T}}\mathbf{a}$ and plug this expression in $J(\mathbf{w})$ so

$$\begin{aligned} J(\mathbf{a}) &= \frac{1}{2}\mathbf{a}^{\mathsf{T}}\Phi\Phi^{\mathsf{T}}\Phi\Phi^{\mathsf{T}}\mathbf{a} - \mathbf{a}^{\mathsf{T}}\Phi\Phi^{\mathsf{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathsf{T}}\mathbf{t} - \frac{\lambda}{2}\mathbf{a}^{\mathsf{T}}\Phi\Phi^{\mathsf{T}}\mathbf{a} \\ &= \frac{1}{2}\mathbf{a}^{\mathsf{T}}KK\mathbf{a} - \mathbf{a}^{\mathsf{T}}K\mathbf{t}\frac{1}{2}\mathbf{t}^{\mathsf{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathsf{T}}K\mathbf{a} \end{aligned}$$

  where $K = \Phi\Phi^{\mathsf{T}}$.
- K is the Gram matrix

$$[K]_{i,j} = \Phi^{\mathsf{T}}(\mathbf{x}_i)\Phi(\mathbf{x}_j)$$

- Note that by construction, K is a p.s.d. matrix; that is $\boldsymbol{\alpha}^{\mathsf{T}}K\alpha \geq \boldsymbol{\alpha}$ for all $\boldsymbol{\alpha}$.

- Solving $\frac{\partial J}{\partial \mathbf{a}} = 0$ yields

$$\mathbf{a} = (K + \lambda I_N)^{-1}\mathbf{t}$$

- It follows that

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathsf{T}}\Phi(\mathbf{x}) = \mathbf{a}^{\mathsf{T}}\Phi\Phi(\mathbf{x}) = k(\mathbf{x})^{\mathsf{T}}(K + \lambda I_N)^{-1}\mathbf{t}$$

  where

$$k(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), ..., k(\mathbf{x}, \mathbf{x}_N))^{\mathsf{T}}$$

- We now have to invert an $N \times N$ matrix instead of an $M \times M$ matrix (where $\Phi(\mathbf{x}) \in \mathbb{R}^M$).
- Now if we let $k(\mathbf{x}, \mathbf{x}')$ be a p.s.d. then you can still define $y(\mathbf{x}, \mathbf{w})$ whereas M is infinite!

Constructing kernels

- Mercer's theorem reformulated: $k(\mathbf{x}, \mathbf{x}')$ is a valid kernel iff the Gram matrix $K = [k(\mathbf{x}_n, \mathbf{x}_m)]$ is positive semi definite for all possible $\{\mathbf{x}_n\}$.
- A matrix A is psd iff $\alpha^\mathsf{T} A \alpha \geq 0$ for all α.
- The corresponding features $\Phi(\cdot)$ are eigenfunctions of k, i.e. $\int k(\mathbf{x}, \mathbf{x}')\Phi_i(\mathbf{x})d\mathbf{x} = \lambda_i \Phi_i(\mathbf{x})$.

Example Kernels

- Stationary: $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$.
- Isotropic: $k(\mathbf{x}, \mathbf{x}') = k(||\mathbf{x} - \mathbf{x}'||)$.
- Monomials of order M: $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^{\mathsf{T}}\mathbf{x}')^M$.
- Monomials of order up to M: $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^{\mathsf{T}}\mathbf{x}' + c)^M$
- "Gaussian" $k(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||^2/2\sigma^2)$.
- Sigmoid "kernel" (does not satisfy Mercer's theorem!): $k(\mathbf{x}, \mathbf{x}') = \tanh(a\mathbf{x}^{\mathsf{T}}\mathbf{x}' + b)$.

Combining Kernels

- Assume $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$ are p.s.d. kernels then we can combine them in multiple ways to obtain new kernels.
- For any $\alpha, \beta > 0$ $k(\mathbf{x}, \mathbf{x}') = \alpha k_1(\mathbf{x}, \mathbf{x}') + \beta k_2(\mathbf{x}, \mathbf{x}')$ is p.s.d.
- $k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}')$ is p.s.d.
- $k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$ is p.s.d.
- $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}')$ is p.s.d.
- $k(\mathbf{x}, \mathbf{x}') = k_1(\Phi(\mathbf{x}), \Phi(\mathbf{x}'))$ is p.s.d.

Gaussian kernel

- The Gaussian kernel $\exp(-||\mathbf{x}-\mathbf{x}'||^2/2\sigma^2)$ might be the most used kernel in practice.
- It is not limited to Euclidean space. Consider that

$$\begin{aligned} ||\mathbf{x}-\mathbf{x}'||^2 &= (\mathbf{x}-\mathbf{x}')^{\mathsf{T}}(\mathbf{x}-\mathbf{x}') \\ &= \mathbf{x}^{\mathsf{T}}\mathbf{x}+\mathbf{x}'^{\mathsf{T}}\mathbf{x}'-2\mathbf{x}^{\mathsf{T}}\mathbf{x}' \end{aligned}$$

  then we can consider a nonlinear kernel where

$$||\mathbf{x}-\mathbf{x}'||^2 \longleftrightarrow k_1(\mathbf{x},\mathbf{x})+k_1(\mathbf{x},\mathbf{x}')-2k_1(\mathbf{x},\mathbf{x}')$$

- We then consider the kernel

$$k(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}\left(k_1(\mathbf{x},\mathbf{x})+k_1(\mathbf{x},\mathbf{x}')-2k_1(\mathbf{x},\mathbf{x}')\right)\right)$$

- Any algorithm where a distance appears can be kernelized...

Kernels on graphs, sets, strings etc

- Over the past few years, there has been a lot of work on defining kernels between non-Euclidean objects.
- The aim is to come up with a p.s.d. kernel.
- It is not though because a kernel is p.s.d. that it is a 'good' measure of similarity.

Kernels derived from probabilistic models

- Generative models (eg HMMs) provide a way to deal with variable-dimension objects (eg strings of different lengths).
- We can then use these for discriminative learning by defining kernels.
- For example for a generative model $p(\mathbf{x})$, we could define

$$k(\mathbf{x}, \mathbf{x}') = p(\mathbf{x})\, p(\mathbf{x}')$$

or

$$k(\mathbf{x}, \mathbf{x}') = \int p(\mathbf{x}|\,\theta)\, p(\mathbf{x}'|\,\theta)\, p(\theta)\, d\theta$$

Fisher Kernel

- Consider a parametric generative model $p(\mathbf{x}|\theta)$.
- We introduce the kernel which uses a feature vector of size $|\theta|$

$$k(\mathbf{x}, \mathbf{x}') = g(\theta, \mathbf{x}) F^{-1} g(\theta, \mathbf{x}')$$

where

$$\begin{aligned} g(\theta, \mathbf{x}) &= \nabla_\theta \log p(\mathbf{x}|\theta) \\ F &= \mathbb{E}_{\mathbf{x}}[g(\theta, \mathbf{x})^{\mathsf{T}} g(\theta, \mathbf{x}')] \end{aligned}$$

- F is the Fisher information matrix, the kernel is invariant to the parametrization of θ.

Gaussian Processes

- A stochastic process is a collection of RVs indexed by the input vector $\mathbf{x}$. A Gaussian Process is a stochastic process for which $(y(\mathbf{x}_1), \ldots, y(\mathbf{x}_n))$ is jointly Gaussian for any $\{\mathbf{x}_n\}$.
- A GP can be characterized by its mean function $m(\mathbf{x})$ (often assumed 0) and its covariance function $k(\mathbf{x}, \mathbf{x}')$; i.e.

$$\mathbb{E}\left[y(\mathbf{x})\right] = m\left(\mathbf{x}\right), \ \ cov\left[y(\mathbf{x}), y(\mathbf{x}')\right] = k(\mathbf{x}, \mathbf{x}')$$

- For any $\{\mathbf{x}_n\}$, we have

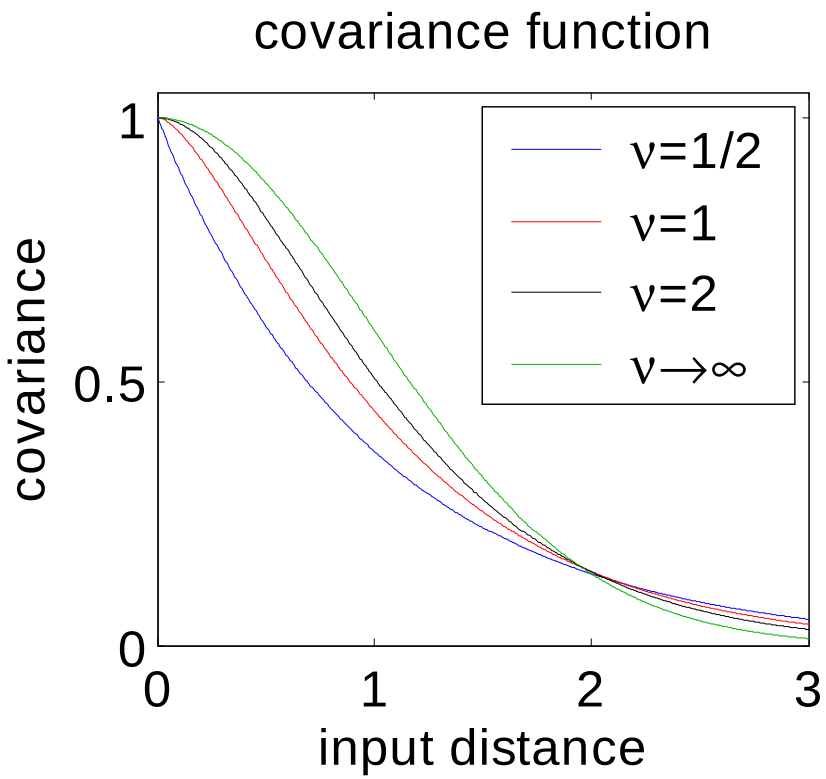
$$y(\mathbf{x}_{1:n}) \sim \mathcal{N}\left(m(\mathbf{x}_{1:n}), K(\mathbf{x}_{1:n})\right)$$

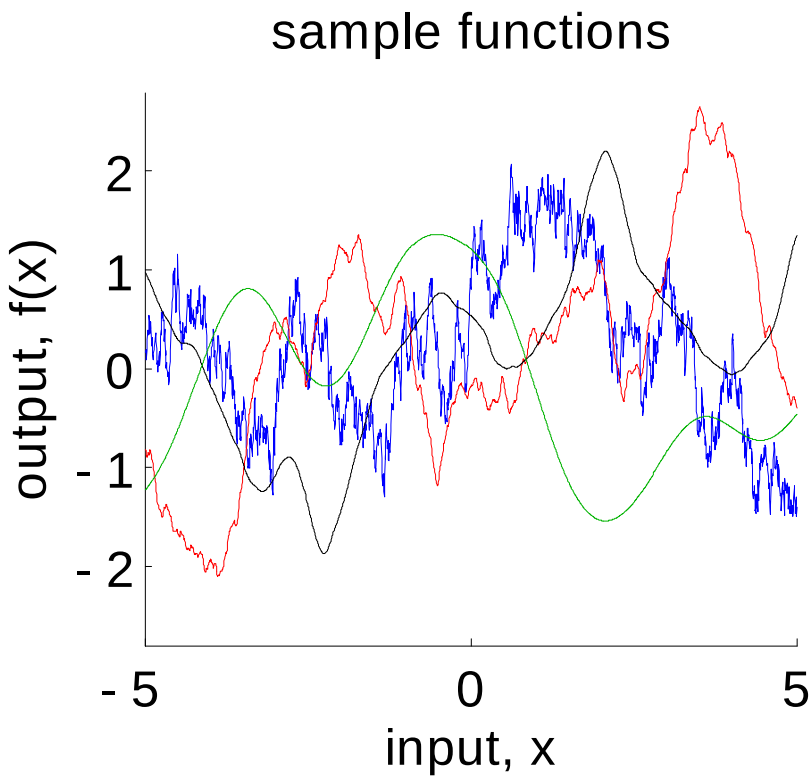
where $y(\mathbf{x}_{1:n}) = (y(\mathbf{x}_1), \ldots, y(\mathbf{x}_n))^{\mathsf{T}}$,
$m(\mathbf{x}_{1:n}) = (m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n))^{\mathsf{T}}$,

$$\left[K(\mathbf{x}_{1:n})\right]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j).$$

- A GP gives a prior on the space of functions.

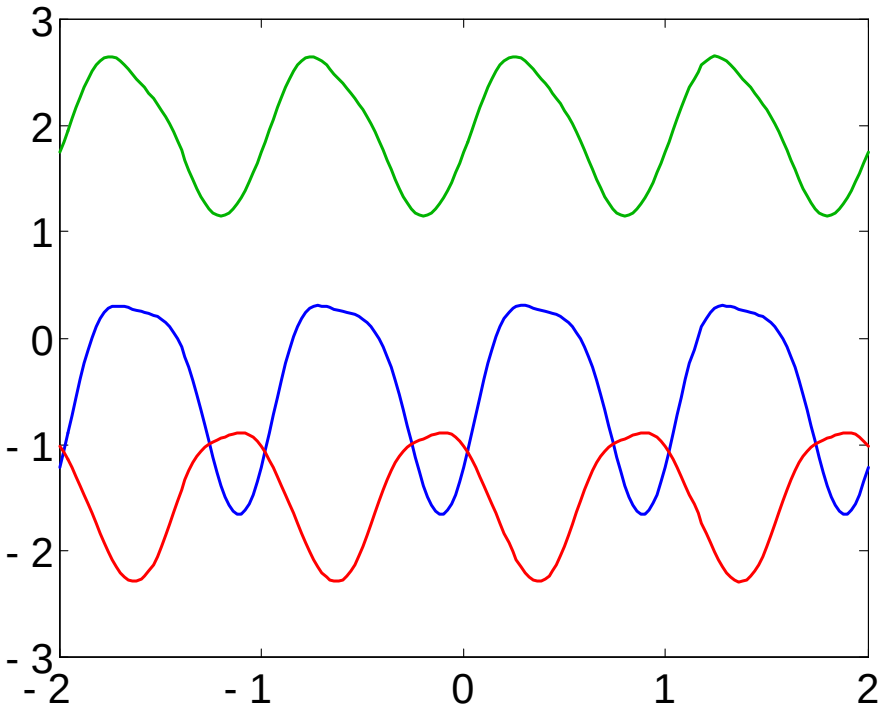
Samples from the prior for Matern covariance

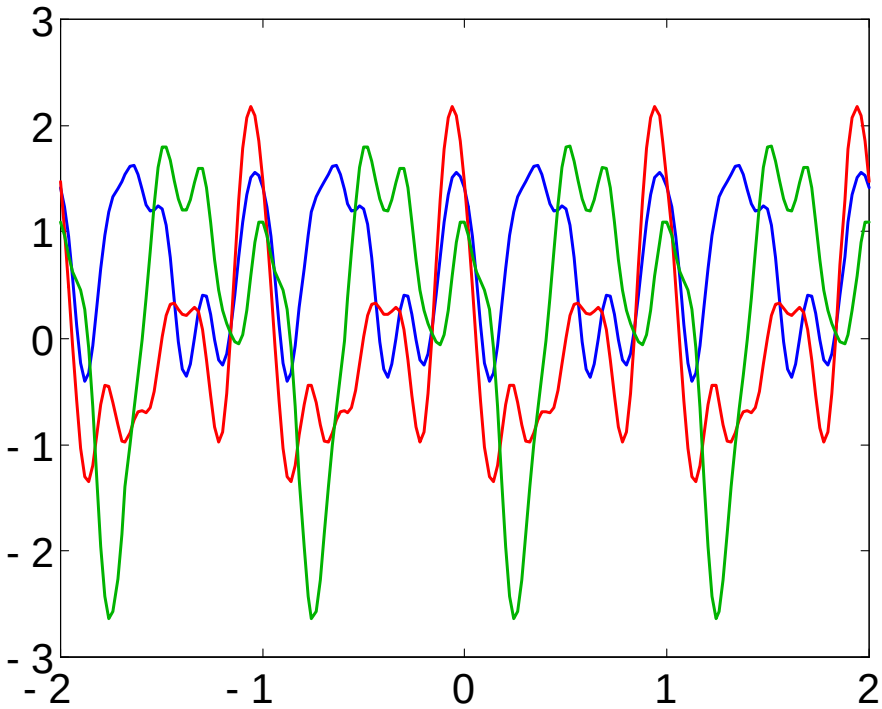




Covariance function $k_\nu(\mathbf{x}, \mathbf{x}') = k_\nu(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{l} \right)^\nu K_\nu \left(\frac{\sqrt{2\nu} r}{l} \right)$ for $\|\mathbf{x} - \mathbf{x}'\| = r$ (left) and sample paths (right)

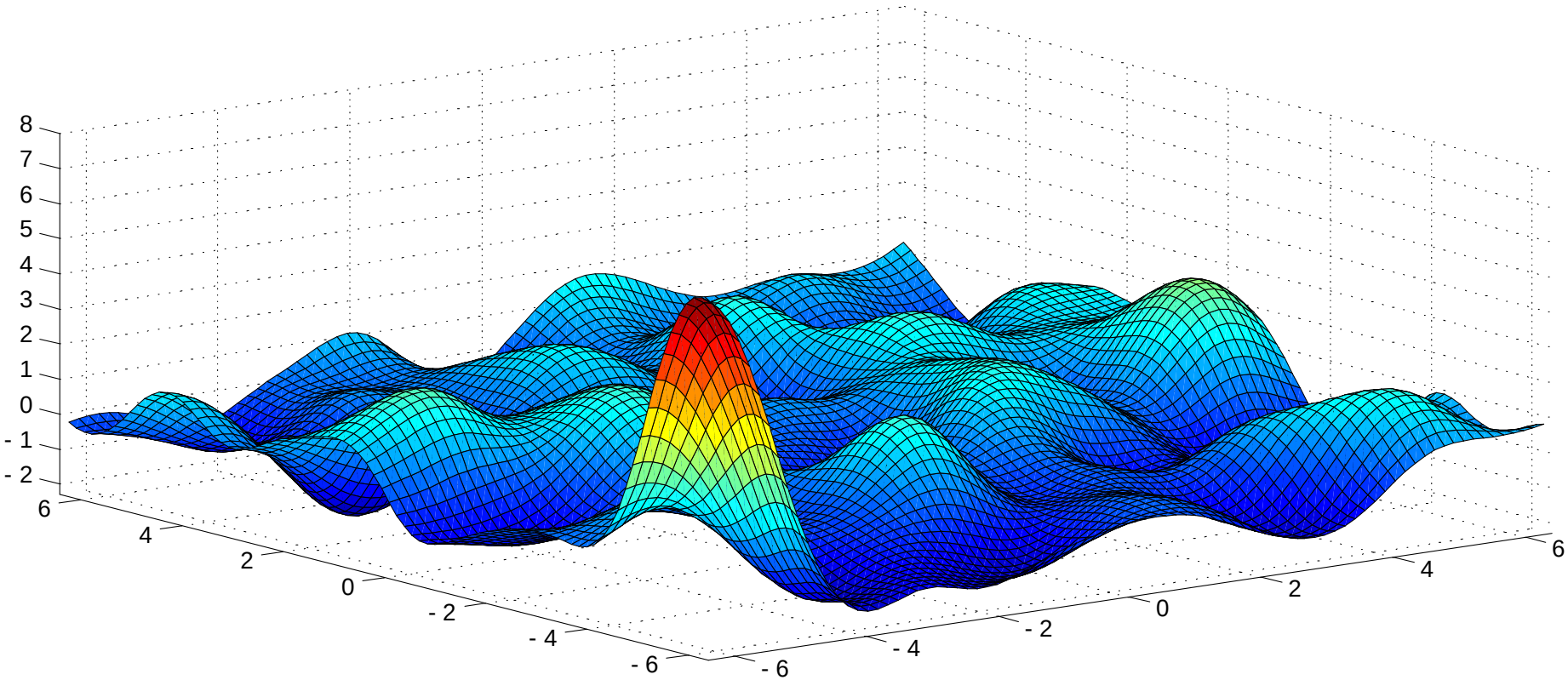
Samples from the prior for a periodic covariance





Sample paths from the prior for $l > 1$ (left) and $l < 1$ (right) where $k_\nu(\mathbf{x}, \mathbf{x}') = \exp\left(-2\sin^2\left(\pi\left(\mathbf{x} - \mathbf{x}'\right)\right)/l^2\right)$

Samples from the prior with a Gaussian covariance



Sample surface for $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\nu^2 \left\|\mathbf{x} - \mathbf{x}'\right\|^2\right)$

Bayesian linear regression & Gaussian Processes

- Consider the linear regression model where

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathsf{T}} \Phi(\mathbf{x})$$

  and we set $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1} I)$.
- $y(\mathbf{x}, \mathbf{w})$ is a linear combination of Gaussians rvs so it is a GP with

$$\mathbb{E}\left[y(\mathbf{x}, \mathbf{w})\right] = \mathbb{E}\left[\mathbf{w}^{\mathsf{T}}\right] \Phi(\mathbf{x}) = 0$$

  and

$$\begin{aligned} cov\left[y(\mathbf{x}, \mathbf{w}), y\left(\mathbf{x}', \mathbf{w}\right)\right] &= \Phi^{\mathsf{T}}(\mathbf{x}) \mathbb{E}\left[\mathbf{w}\mathbf{w}^{\mathsf{T}}\right] \Phi(\mathbf{x}) \\ &= \alpha^{-1} \Phi^{\mathsf{T}}(\mathbf{x}) \Phi(\mathbf{x}'). \end{aligned}$$

- Instead of introducing a prior on $y(\mathbf{x})$ by defining a prior on $\mathbf{w}$ and introducing a finite dimensional vector of features, we can directly introduce a GP prior on $y(\mathbf{x})$.

Bayesian regression with Gaussian Processes

- Consider the data $D = \{\mathbf{x}_n, t_n\}_{n=1}^N$ where

$$t_n = t(\mathbf{x}_n) = y(\mathbf{x}_n) + \varepsilon_n \text{ where } \varepsilon_n \sim \mathcal{N}(0, \sigma^2)$$

and

$$y(\mathbf{x}) \sim GP\left(m(\mathbf{x}) = 0, k\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

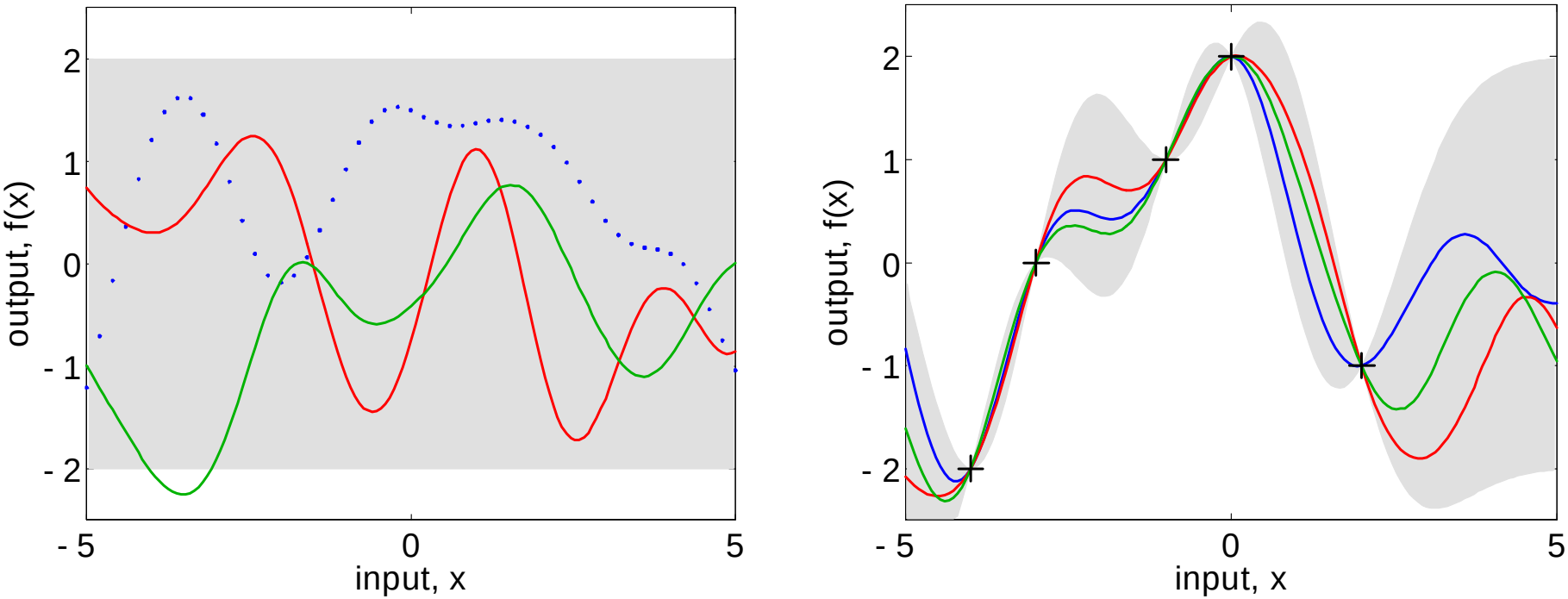
- We have

$$y(\mathbf{x}) | D \sim GP\left(m_{\text{post}}(\mathbf{x}), k_{\text{post}}\left(\mathbf{x}, \mathbf{x}'\right)\right)$$

where

$$\begin{aligned} m_{\text{post}}(\mathbf{x}) &= k(\mathbf{x}, \mathbf{x}_{1:N}) \left[K(\mathbf{x}_{1:N}, \mathbf{x}_{1:N}) + \sigma^2 I\right]^{-1} \mathbf{t}_{1:N}, \\ k_{\text{post}}\left(\mathbf{x}, \mathbf{x}'\right) &= k\left(\mathbf{x}, \mathbf{x}'\right) - \\ & \quad k(\mathbf{x}, \mathbf{x}_{1:N}) \left[K(\mathbf{x}_{1:N}, \mathbf{x}_{1:N}) + \sigma^2 I\right]^{-1} k\left(\mathbf{x}_{1:N}, \mathbf{x}'\right) \end{aligned}$$

From the prior to the posterior



Random draws from the prior (left) and the posterior (right): The shaded area represents the pointwise mean +/- twice the standard deviation.

Predictive distribution and Interpretation

- Given $\mathbf{x}^*$, we have

$$p\left(t^* \middle| \mathbf{x}_{1:N}, \mathbf{t}_{1:N}, \mathbf{x}^*\right) = \mathcal{N}\left(t^*; \mu\left(\mathbf{x}^*\right), \sigma^2\left(\mathbf{x}^*\right)\right)$$

where

$$\begin{aligned}
\mu\left(\mathbf{x}^*\right) &= k\left(\mathbf{x}^*, \mathbf{x}_{1:N}\right)\left[K\left(\mathbf{x}_{1:N}, \mathbf{x}_{1:N}\right) + \sigma^2 I\right]^{-1} \mathbf{t}_{1:N}, \\
\sigma^2\left(\mathbf{x}^*\right) &= k\left(\mathbf{x}^*, \mathbf{x}^*\right) + \sigma^2 \\
&\quad - k\left(\mathbf{x}^*, \mathbf{x}_{1:N}\right)\left[K\left(\mathbf{x}_{1:N}, \mathbf{x}_{1:N}\right) + \sigma^2 I\right]^{-1} k\left(\mathbf{x}_{1:N}, \mathbf{x}^*\right)
\end{aligned}$$

- The mean $\mu\left(\mathbf{x}^*\right)$ is linear in two ways

$$\mu\left(\mathbf{x}^*\right) = \sum_{i=1}^{n} a_i t_i = \sum_{i=1}^{n} b_i K\left(\mathbf{x}^*, \mathbf{x}_i\right)$$

- The variance is of the form

$$\sigma^2\left(\mathbf{x}^*\right) = \text{prior variance - positive terms dependent on } \mathbf{x}_{1:N}$$

- *Remark*: the variance is independent of the observations $\mathbf{t}_{1:N}$.

Computational Complexity

- The central computation operation in using GP involves inverting a $N \times N$ matrix. Standard methods requires $O\left(N^3\right)$ operations.
- In the finite basis function model with M basis, we have to invert a $M \times M$ matrix.
- So if the number M of basis functions is smaller than N then we are better off with the standard method.
- If the kernel considered corresponds to an infinite M, we do not have the choice!
- Several techniques have been developed to perform approximate inference.

Learning the hyperparameters

- In practice, we often parametrize the kernel by some parameters θ.
- To estimate θ, we can maximize the marginal log-likelihood

$$\log p\left(\mathbf{t}_{1:N} \middle| \theta, \mathbf{x}_{1:N}\right) = -\frac{1}{2}\log\left|K_N^\theta\right| - \frac{1}{2}\mathbf{t}_{1:N}^\mathsf{T}\left[K_N^\theta\right]^{-1}\mathbf{t}_{1:N} - \frac{N}{2}\log 2\pi$$

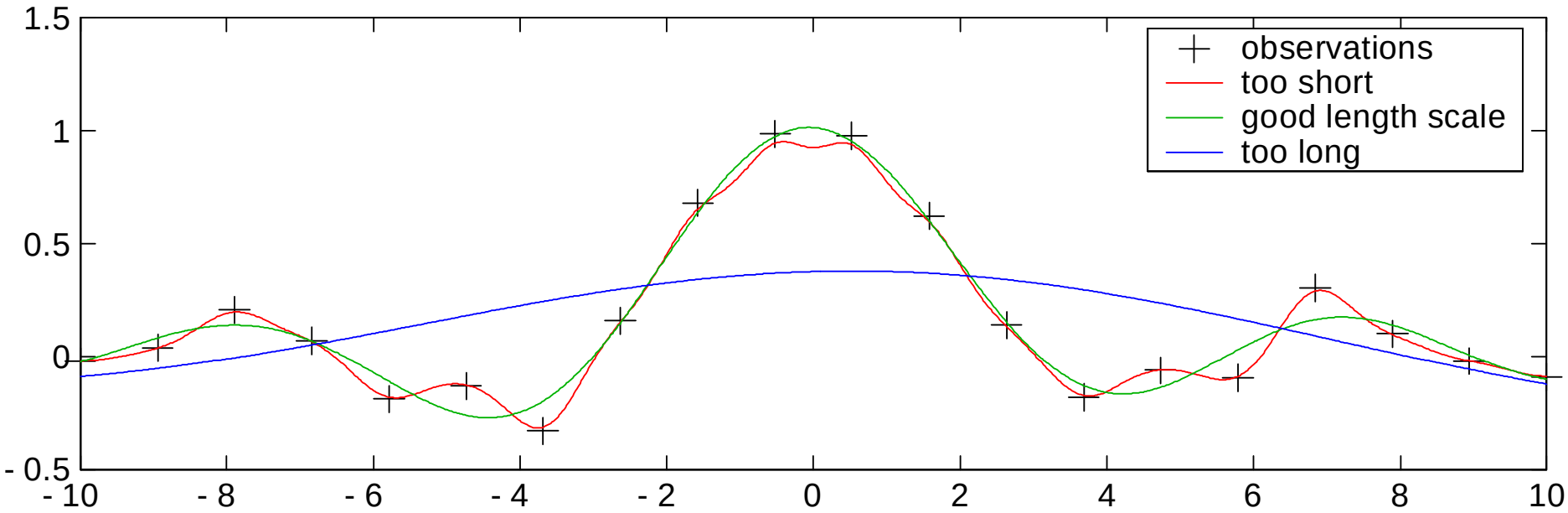
  using $\left[K_N^\theta\right]_{i,j} = K^\theta\left(\mathbf{x}_i, \mathbf{x}_j\right)$.
- The gradient of the log-likelihood is given by

$$\begin{aligned}\frac{\partial \log p\left(\mathbf{t}_{1:N} \middle| \theta, \mathbf{x}_{1:N}\right)}{\partial \theta_i} &= -\frac{1}{2}\mathrm{Tr}\left(\left[K_N^\theta\right]^{-1}\frac{\partial K_N^\theta}{\partial \theta_i}\right) \\ &\quad + \frac{1}{2}\mathbf{t}_{1:N}^\mathsf{T}\left[K_N^\theta\right]^{-1}\frac{\partial K_N^\theta}{\partial \theta_i}\left[K_N^\theta\right]^{-1}\mathbf{t}_{1:N}\end{aligned}$$

- The log-likelihood is typically not concave in θ.

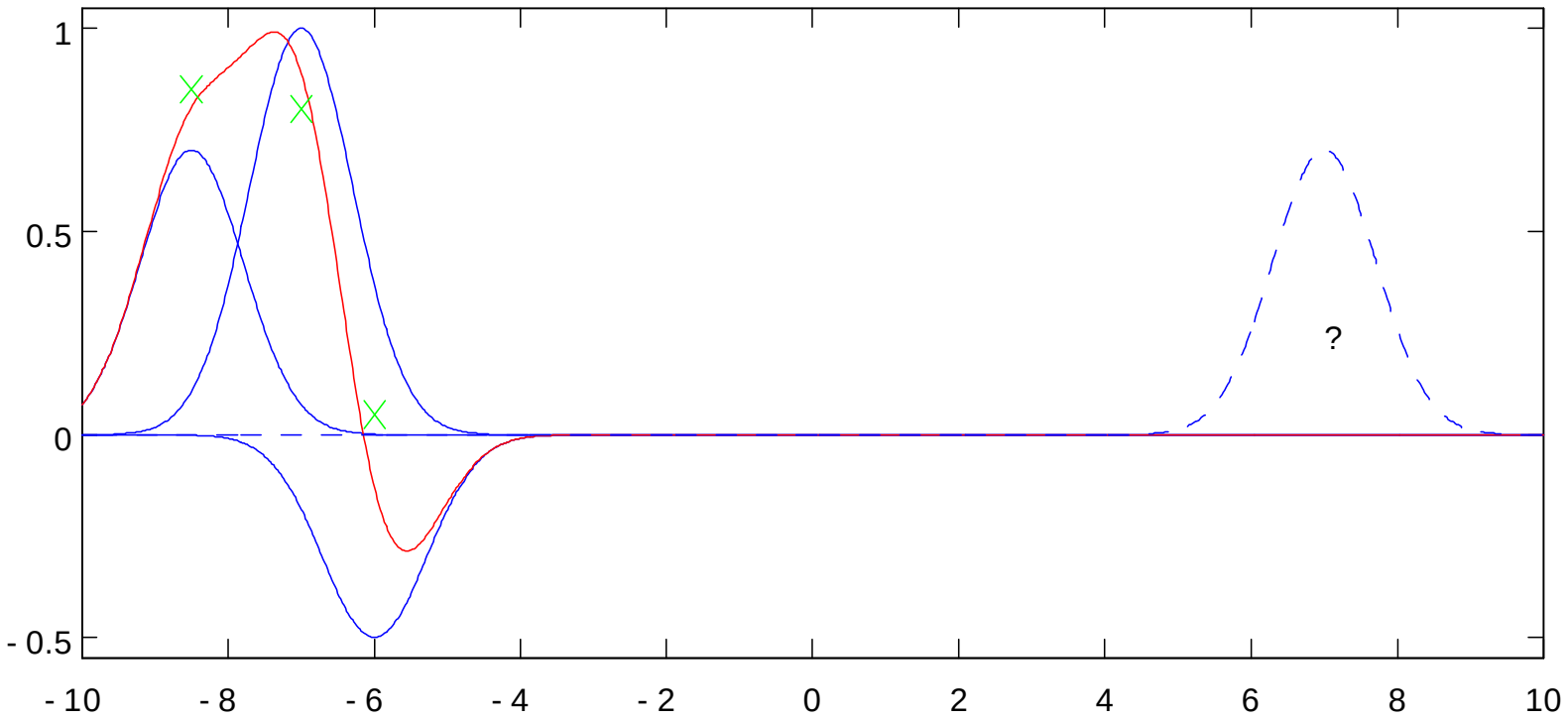
Example: Fitting the length scale parameter

- Parameterized covariance function: $k(\mathbf{x}, \mathbf{x}') = \nu \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{l}\right)$.



- The mean posterior predictive distribution is plotted for 3 different length scales (the green curve corresponds to optimizing the likelihood). Note that we can get an almost perfect fit for a small length scale but the marginal likelihood does not favour it.

Using a finite number of basis functions can be dangerous

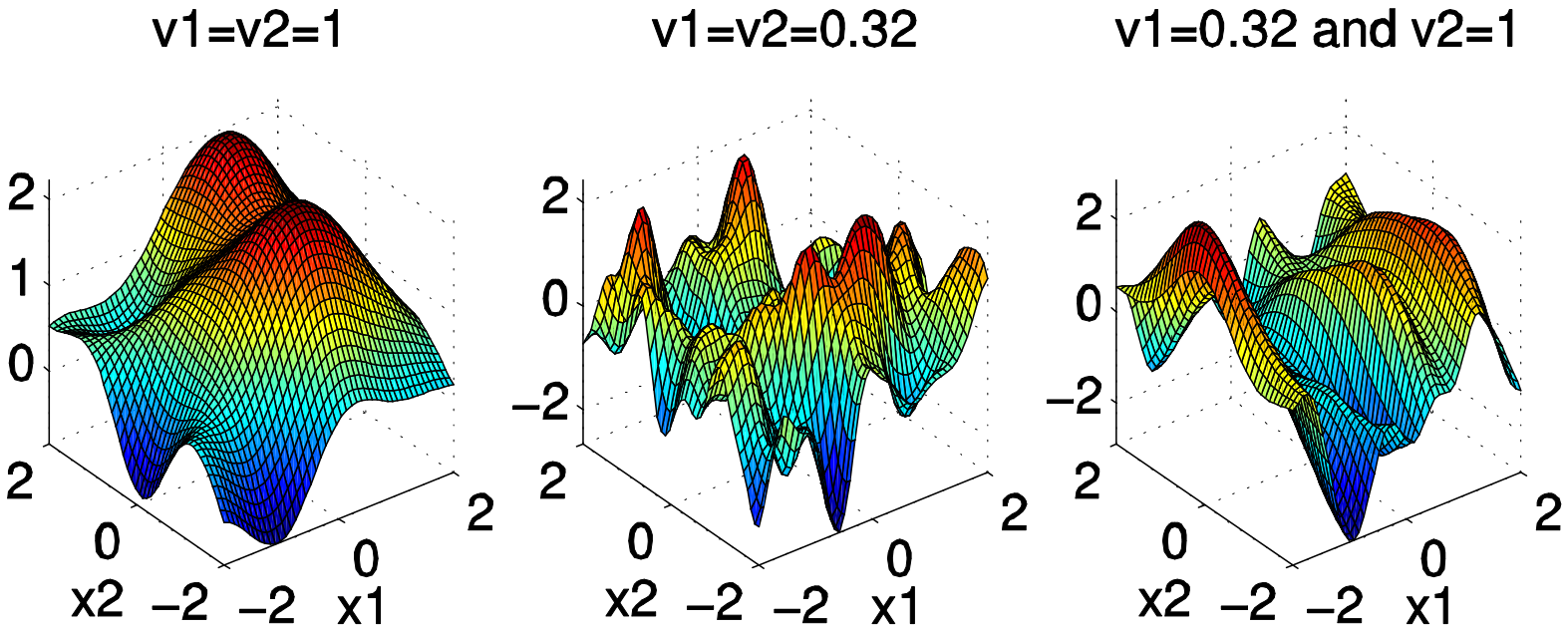


Automatic Relevance Determination

- We can extend the technique described before to select automatically the relevant input variables; i.e. say

$$k\left(\mathbf{x},\mathbf{x}'\right)=\nu_0^2\exp\left(-\frac{\sum_{i=1}^{D}\left(x_i-x_i'\right)^2}{2\nu_i^2}\right)$$

  where $\theta=\left(\nu_0^2,\nu_1^2,...,\nu_D^2\right)$.
- We have



Gaussian Processes for Binary Classfication

- The input is given by $\mathbf{x}$ and the output $t \in \{0, 1\}$ with

$$\Pr(t = 1 | \mathbf{x}) = g(a(\mathbf{x})).$$

- We model $a(\mathbf{x})$ through a Gaussian process define by

$$\mathbb{E}[a(\mathbf{x})] = 0 \text{ and } cov[a(\mathbf{x}) a(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') = m(\mathbf{x}, \mathbf{x}') + \nu\delta(\mathbf{x} - \mathbf{x}')$$

- We are interested in computing

$$\begin{aligned} p(t^* | \mathbf{x}_{1:N}, \mathbf{t}_{1:N}, \mathbf{x}^*) &= \int p(t^* | a(\mathbf{x}^*)) p(a(\mathbf{x}^*) | \mathbf{t}_{1:N}) da(\mathbf{x}^*) \\ &= \int g(a(\mathbf{x}^*)) p(a(\mathbf{x}^*) | \mathbf{t}_{1:N}) da(\mathbf{x}^*) \end{aligned}$$

Laplace Approximation

- We have

$$\begin{aligned} p\left(a\left(\mathbf{x}^{*}\right)\middle|\,\mathbf{t}_{1:N}\right) &= \int p\left(a\left(\mathbf{x}^{*}\right), a\left(\mathbf{x}_{1:N}\right)\middle|\,\mathbf{t}_{1:N}\right) da\left(\mathbf{x}_{1:N}\right) \\ &= \int p\left(a\left(\mathbf{x}^{*}\right)\middle|\,a\left(\mathbf{x}_{1:N}\right)\right) p\left(a\left(\mathbf{x}_{1:N}\right)\middle|\,\mathbf{t}_{1:N}\right) da\left(\mathbf{x}_{1:N}\right) \end{aligned}$$

- We have

$$\begin{aligned} p\left(a\left(\mathbf{x}^{*}\right)\middle|\,a\left(\mathbf{x}_{1:N}\right)\right) = \mathcal{N}\left(a\left(\mathbf{x}^{*}\right); k^{\mathsf{T}}\left(\mathbf{x}^{*}, \mathbf{x}_{1:N}\right) K_N^{-1} a\left(\mathbf{x}_{1:N}\right),\right. \\ \left. k\left(\mathbf{x}, \mathbf{x}\right) - k^{\mathsf{T}}\left(\mathbf{x}^{*}, \mathbf{x}_{1:N}\right) K_N^{-1} k\left(\mathbf{x}^{*}, \mathbf{x}_{1:N}\right)\right) \end{aligned}$$

- We make a Gaussian approximation of $p\left(a\left(\mathbf{x}_{1:N}\right)\middle|\,\mathbf{t}_{1:N}\right)$ using Laplace.

- The unnormalized posterior is given by

$$\begin{aligned} & \log p\left(a\left(\mathbf{x}_{1:N}\right), \mathbf{t}_{1:N}\right) \\ = & \log p\left(a\left(\mathbf{x}_{1:N}\right), \mathbf{t}_{1:N}\right) + \log p\left(\mathbf{t}_{1:N} \middle| a\left(\mathbf{x}_{1:N}\right)\right) \\ = & -\frac{1}{2} a^{\mathsf{T}}\left(\mathbf{x}_{1:N}\right) K_N^{-1} a\left(\mathbf{x}_{1:N}\right) - \frac{N}{2} \log\left(2\pi\right) - \frac{1}{2} \log\left|K_N\right| \\ & + \mathbf{t}_{1:N}^{\mathsf{T}} a\left(\mathbf{x}_{1:N}\right) - \sum_{n=1}^{N} \log\left(1 + \exp a\left(\mathbf{x}_N\right)\right) + cst \end{aligned}$$

  as $g\left(a\right)^t \left(1 - g\left(a\right)\right)^{1-t} = \exp\left(at\right) g\left(-a\right)$
- We perform a Taylor expansion of the $\log p\left(a\left(\mathbf{x}_{1:N}\right), \mathbf{t}_{1:N}\right)$ around its mode which can be computed using a Newton-Raphson method where

$$\nabla \log p\left(a\left(\mathbf{x}_{1:N}\right), \mathbf{t}_{1:N}\right) = \mathbf{t}_{1:N} - \sigma_{1:N} - K_N^{-1} a\left(\mathbf{x}_{1:N}\right)$$

  and

$$\nabla\nabla \log p\left(a\left(\mathbf{x}_{1:N}\right), \mathbf{t}_{1:N}\right) = -W_N - K_N^{-1}$$

  where $W_N = \text{diag}(g\left(a\left(\mathbf{x}_N\right)\right)\left(1 - g\left(a\left(\mathbf{x}_N\right)\right)\right))$.

- The Newton-Raphson formula takes the form

$$a^{(k+1)}\left(\mathbf{x}_{1:N}\right) = K_N\left(I + W_N K_N\right)^{-1}\left\{\mathbf{t}_{1:N} - \sigma_{1:N} + W_N a\left(\mathbf{x}_{1:N}\right)\right\}$$

- Once the mode $a^*\left(\mathbf{x}_{1:N}\right)$ has been found, we compute the associated

$$H = -\nabla\nabla \log p\left(a\left(\mathbf{x}_{1:N}\right), \mathbf{t}_{1:N}\right) = W_N + K_N^{-1}$$

- The Gaussian approximation is given by

$$q\left(a\left(\mathbf{x}_{1:N}\right)\right) = \mathcal{N}\left(a\left(\mathbf{x}_{1:N}\right); a^*\left(\mathbf{x}_{1:N}\right), H\right)$$

- It follows that we obtain a Gaussian approximation of $p\left(a\left(\mathbf{x}^*\right)\middle|\, \mathbf{t}_{1:N}\right)$ with

$$\begin{aligned}
\mathbb{E}\left(a\left(\mathbf{x}^*\right)\middle|\, \mathbf{t}_{1:N}\right) &= k\left(\mathbf{x}^*, \mathbf{x}_{1:N}\right)\left(\mathbf{t}_{1:N} - \boldsymbol{\mu}_{1:N}\right), \\
\mathbb{V}\left(a\left(\mathbf{x}^*\right)\middle|\, \mathbf{t}_{1:N}\right) &= k\left(\mathbf{x}^*, \mathbf{x}^*\right) \\
&\quad - k^{\mathrm{T}}\left(\mathbf{x}^*, \mathbf{x}_{1:N}\right)\left(W_N^{-1} + K_N\right)^{-1} k\left(\mathbf{x}^*, \mathbf{x}_{1:N}\right)
\end{aligned}$$

- Now we finally use the approximation combining logistic and Gaussian

$$p\left(t^* \middle| \mathbf{x}_{1:N}, \mathbf{t}_{1:N}, \mathbf{x}^*\right) = \int g\left(a\left(\mathbf{x}^*\right)\right) p\left(a\left(\mathbf{x}^*\right) \middle| \mathbf{t}_{1:N}\right) da\left(\mathbf{x}^*\right)$$

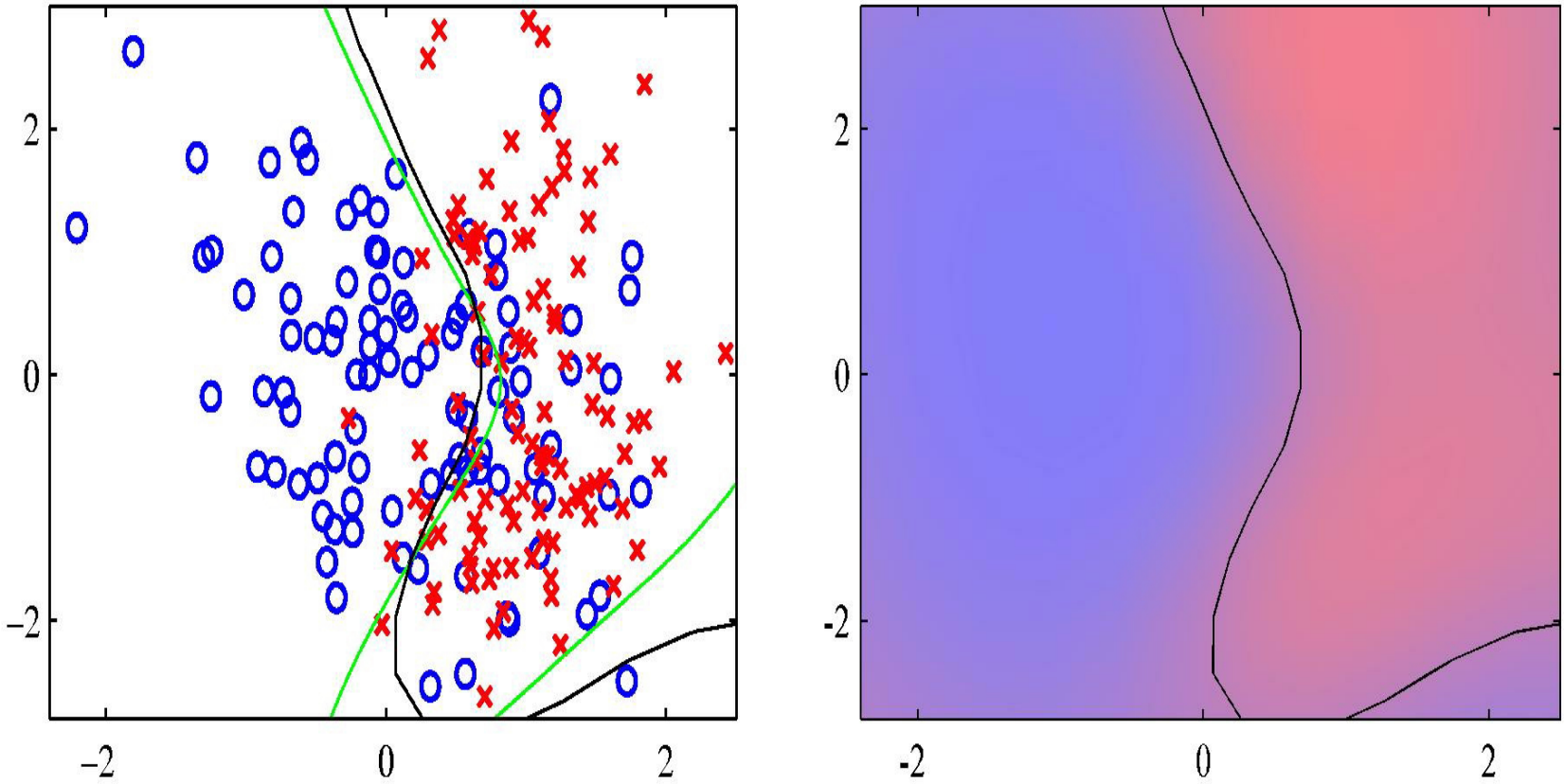
  which states that

$$\int g\left(a\right) \mathcal{N}\left(a; \mu, \sigma^2\right) da \simeq g\left(\frac{\mu}{\sqrt{1 + \pi\sigma^2/8}}\right)$$

- The Laplace approximation also yields an approximation of the log-marginal likelihood

$$\log p\left(\mathbf{t}_{1:N}\right) \simeq \log p\left(a^*\left(\mathbf{x}_{1:N}\right), \mathbf{t}_{1:N}\right) - \frac{1}{2}\left|H\right| + \frac{N}{2}\log\left(2\pi\right)$$

Example of Binary Classification using GP



Left: Optimal decision boundary (green) and GP classifier (black).
Right: predicted posterior proba for the blue and red classes