HELSINKI UNIVERSITY OF TECHNOLOGY
NEURAL NETWORKS RESEARCH CENTRE

D.J.C. MacKay: Information theory, inference and learning algorithms

Ch. 44: Supervised Learning in Multilayer Networks
Ch. 45: Gaussian Processes

Additional material from: Schölkopf, Smola: *Learning with Kernels*

presented by Jarkko Salojärvi, 29.4.2004

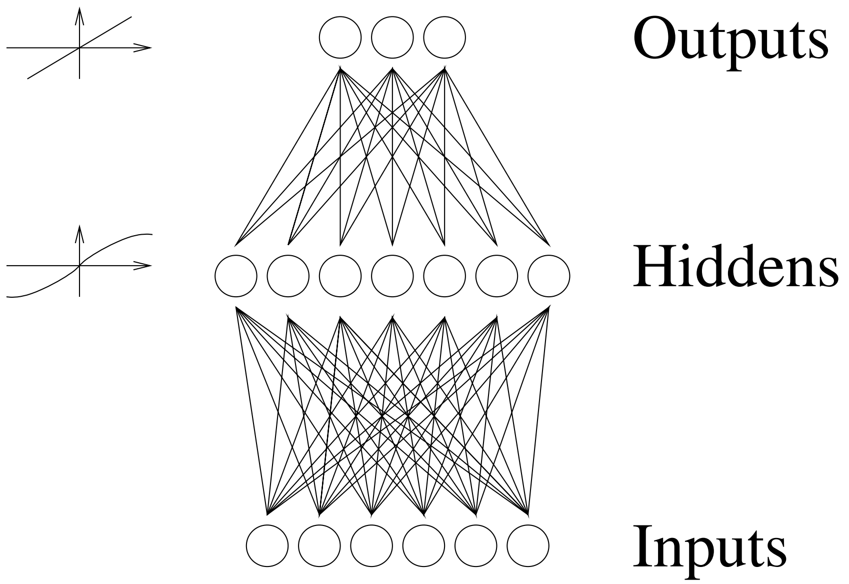
Structure of the presentation

- MLP (briefly)
- Gaussian Processes
  - Definition
  - Implementation
  - Some examples
  - Summary
- Exercises

MLP

Output layer:
$a_j^{(2)} = \sum_j w_{ij}^{(2)} h_j + \Theta_i^{(2)}$
$y_i = f^{(2)}(a_i^{(2)})$
$f^{(2)}$ linear in regression; softmax in classification

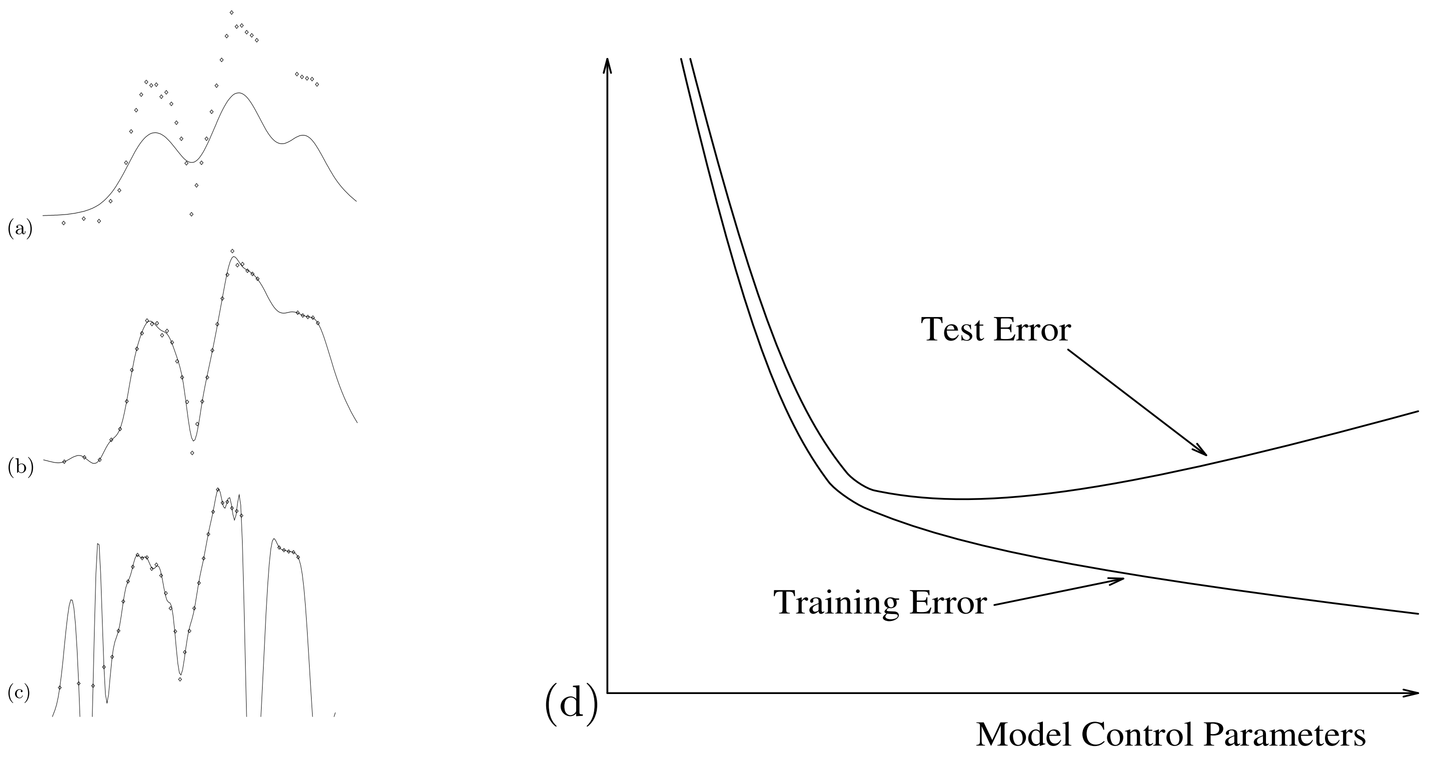
Hidden layer:
$a_j^{(1)} = \sum_l w_{jl}^{(1)} x_l + \Theta_j^{(1)}$
$h_j = f^{(1)}(a_j^{(1)})$
$f^{(1)}$ nonlinear, e.g. $\tanh$ or sigmoid.



MLP for regression

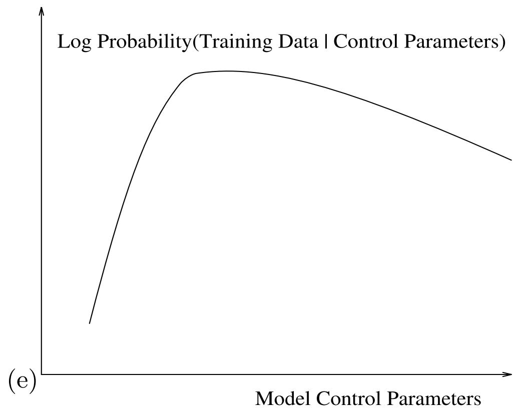
- Error: $E_D(\boldsymbol{w}) = \frac{1}{2}\sum_n \sum_i \left(t_i^{(n)} - y_i(\boldsymbol{x}^{(n)}; \boldsymbol{w})\right)^2$
- Regularization, e.g. $E_W = \frac{1}{2}\sum_i w_i^2$
- Objective function: $M(\boldsymbol{w}) = \beta E_D + \alpha E_W$
- Probabilistic interpretation:
  - $\beta E_D(\boldsymbol{w})$ is the $-\log$ likelihood of a noise model.
  - αE_W is the $-\log$ prior probability of weights
  - Find the posterior $p(\boldsymbol{w}|D, \alpha, \beta) = \frac{1}{Z_M}\exp\{-M(\boldsymbol{w})\}$.

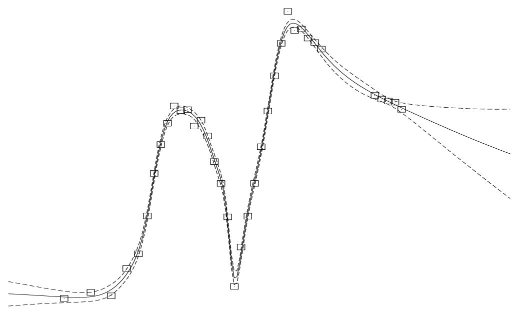
On the importance of regularization



- Models without regularization tend to *overfit* the data.

Benefits of Bayesian methods





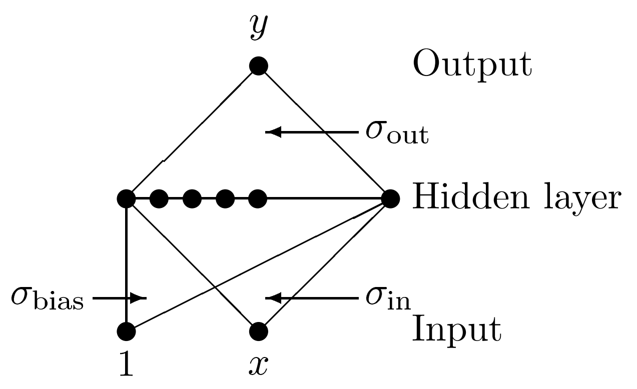
Selection criteria for model complexity (Evidence Z_M)

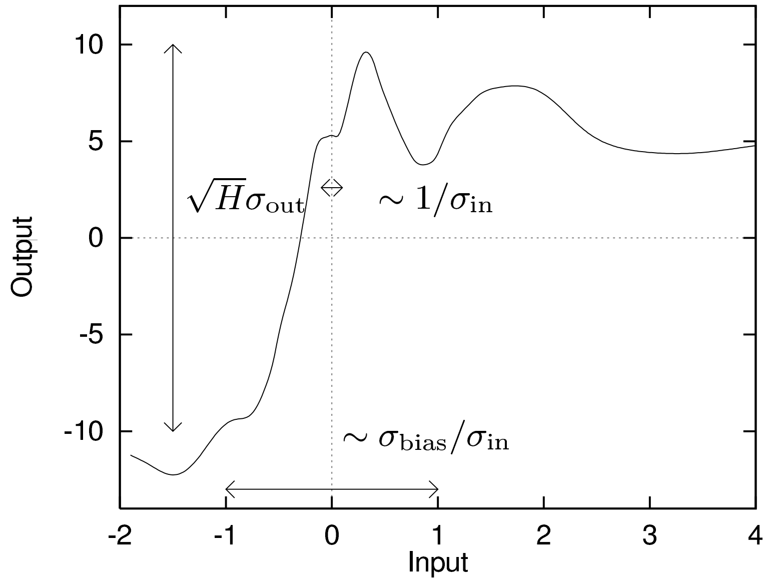
Error bars from posterior pdf

- Validation data set is not needed.
- Regularization constants can be optimized on-line.
- Evidence is not noisy (cf. CV), and its gradient can be evaluated.
- Feature selection with Automatic Relevance Determination (ARD) prior.

Functions produced by random network

How does the output of (a large) MLP behave if its weights are random samples from Gaussian distribution?

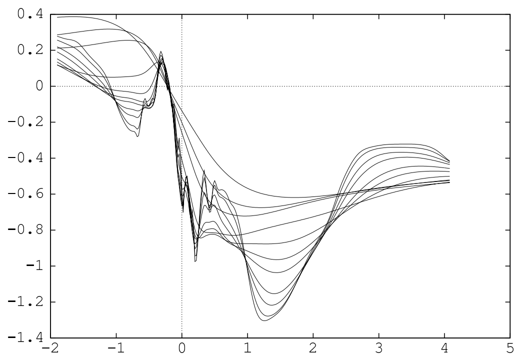




H number of hidden nodes	σ_{out} stdev. of weights in output layer
σ_{in} stdev. of weights in input layer	σ_{bias} stdev. of biases in input layer

...random network

The less we regularize (i.e. larger σ), the more complex functions we will get.



- As $H \to \infty$ the complexity of the functions becomes independent of the number of parameters in the model.
- MLP with one hidden layer and Gaussian priors for weights $\Rightarrow$ *Gaussian process* as $H \to \infty$.

Bayesian view on regression

MLPs are universal function approximators.

From a Bayesian viewpoint we are trying to compute the posterior distribution of a function $y(\boldsymbol{x})$:

$$P(y(\boldsymbol{x})|\boldsymbol{t}_N, \boldsymbol{X}_N) = \frac{P(\boldsymbol{t}_N|y(\boldsymbol{x}), \boldsymbol{X}_N)P(y(\boldsymbol{x}))}{P(\boldsymbol{t}_N|\boldsymbol{X}_N)}.$$

$P(y(\boldsymbol{x}))$ is a prior on the functions. It is often implicit, since priors are usually placed on parameters of the function approximator.
Desired priors on parameteres are such that $y(\boldsymbol{x})$ is continuous and smooth, and has less high frequency power than low frequency power.

Gaussian Process

Process modelling: Place the prior $P(y(\boldsymbol{x}))$ directly on the space of functions instead of on the parameters of y.

Gaussian Process: The prior has a Gaussian form,
$P(y(\boldsymbol{x})) = \frac{1}{Z} \exp\left\{-\frac{1}{2} y(\boldsymbol{x})^T A y(\boldsymbol{x})\ \right\}$.

GPs can be seen as a generalization of Gaussian probability distribution to space of functions.

- GP is specified by mean and covariance functions, $\mu(\boldsymbol{x})$ and $C(\boldsymbol{x}, \boldsymbol{x}')$.
- One sample from GP prior is a function $y(\boldsymbol{x})$.

Gaussian Process: Definition

Probability distribution of a function $y(\boldsymbol{x})$ is a Gaussian process if for any finite selection of points $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}$, the density $P(y^{(1)}, y^{(2)}, \ldots, y^{(N)})$ is Gaussian.

GP: Example - Regression

Data: $\boldsymbol{X}_N, \boldsymbol{t}_N = \{\boldsymbol{x}^{(n)}, t_n\}_{n=1}^{N}$
Task: Predict t_{N+1}, given $\boldsymbol{x}^{(N+1)}$

Model 1 (parametric): Model t with $y(\boldsymbol{x};\boldsymbol{w}) = \sum_{h=1}^{H} w_h \phi_h(\boldsymbol{x})$, where $\phi_h(\boldsymbol{x}) = \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{c}_h)}{2r^2}\right]$ are a set of radial basis functions centered at fixed points $\{\boldsymbol{c}_h\}_{h=1}^{H}$.

Model 2 (non-parametric): Model t with cubic splines, i.e. select $\hat{y}(\boldsymbol{x})$ that minimizes

$$M(y(x)) = \frac{1}{2}\beta \sum_n (y(x^{(n)}) - t_n)^2 + \frac{1}{2}\alpha \int [y^{(p)}(x)]^2 dx,$$

where p=2 (second derivative).

Model 1 - Parametric

Place prior on $\boldsymbol{w}$ and compute the posterior distribution $p(\boldsymbol{w}|\boldsymbol{X}_N, \boldsymbol{t}_N)$ (using MCMC or Laplace).

For prediction we need to compute

$p(t_{N+1}|\boldsymbol{X}_{N+1}, \boldsymbol{t}_N) = \int P(t_{N+1}|\boldsymbol{w}, \boldsymbol{x}^{(N+1)})P(\boldsymbol{w}|\boldsymbol{t}_N, \boldsymbol{X}_N)d^H\boldsymbol{w}.$

It seems that the way in which $y(\boldsymbol{x})$ is represented is not relevant.

Model 2 - Non-parametric

Probabilistic interpretation for the cost function $M(y(x))$:

Likelihood: $\log P(\boldsymbol{t}_N | y(x), \beta) = -\frac{1}{2}\beta \sum_n (y(x^{(n)}) - t_n)^2 + \text{const.}$

Prior: $\log P(y(x)|\alpha) = -\frac{1}{2}\alpha \int [y^{(p)}(x)]^2 dx + \text{const.}$

Splines can be written as parametric models

Use Fourier transform: $y(x) = \sum_h w_{h(\cos)} \cos(hx) + \sum_h w_{h(\sin)} \sin(hx)$

Use regularizer $E_W(\boldsymbol{w}) = \sum_h \frac{1}{2} h^{\frac{p}{2}} w_{h(\cos)}^2 + \frac{1}{2} h^{\frac{p}{2}} w_{h(\sin)}^2$

Splines priors are Gaussian processes

The prior can be written as $-\frac{1}{2}\alpha \int [y^{(p)}(x)]^2 dx =$
$-\frac{1}{2}\alpha \int [y(x)^T D^{(p)^T}][D^{(p)} y(x)] dx = -\frac{1}{2} y(x)^T A y(x)$ (GP prior)

Model 1: from RBF to GP

Since splines can be written in terms of basis functions, we will concentrate on model 1 from now on.

Define $\boldsymbol{R}$ to be a $N \times H$ matrix of values of H basis functions at points $\{\boldsymbol{x}_N\}$, $R_{nh} = \phi_h(\boldsymbol{x}^{(n)})$.

Define vector $\boldsymbol{y}_n = \sum_h R_{nh} w_h$

Assume $P(\boldsymbol{w}) = \mathcal{N}(0, \sigma_w^2 \boldsymbol{I})$

We will next compute the covariance of t, and then inspect its properties when $H \to \infty$. It will turn out that the covariance depends only on $\{\boldsymbol{x}_N\}$.

Model 1: Covariances

Covariance of $\boldsymbol{y}$:

$$\boldsymbol{Q} = E_w\left[\boldsymbol{y}\boldsymbol{y}^T\right] = E_w\left[\boldsymbol{R}\boldsymbol{w}\boldsymbol{w}^T\boldsymbol{R}^T\right] = \sigma_w^2 \boldsymbol{R}\boldsymbol{R}^T.$$

$\Rightarrow$ The prior for $\boldsymbol{y}$ is Gaussian $P(\boldsymbol{y}) = \mathcal{N}(\mathbf{0}, \boldsymbol{Q})$

- If $H < N$, $\boldsymbol{Q}$ does not have full rank.

Covariance of $\boldsymbol{t}$:

Observations $t_n = y(x^{(n)}) + \nu$, where $\nu \sim \mathcal{N}(\mathbf{0}, \sigma_\nu^2 \boldsymbol{I})$ is additive Gaussian noise.

$\Rightarrow$ The prior for $\boldsymbol{t}$ is Gaussian $P(\boldsymbol{t}) = \mathcal{N}(\mathbf{0}, \boldsymbol{Q} + \sigma_\nu^2 \boldsymbol{I})$.

- The covariance $\boldsymbol{C} = \boldsymbol{Q} + \sigma_\nu^2 \boldsymbol{I}$ is full rank.

Model 1: $H \to \infty$

Each entry in $Q_{nn'} = \sigma_w^2 \sum_h \phi_h(\boldsymbol{x}^{(n)})\phi_h(\boldsymbol{x}^{(n')})$. Assume that σ_w^2 scales as S/dh. Then, assuming that hth basis function is centered at h

$$Q_{nn'} = S \int_{h_{\min}}^{h_{\max}} \phi_h(\boldsymbol{x}^{(n)})\phi_h(\boldsymbol{x}^{(n')}) dh = S \int_{h_{\min}}^{h_{\max}} e^{-\frac{(\boldsymbol{x}^{(n)}-h)^2}{2r^2}} e^{-\frac{(\boldsymbol{x}^{(n')}-h)^2}{2r^2}} dh$$

$$= S e^{-\frac{(\boldsymbol{x}^{(n)}-\boldsymbol{x}^{(n')})^2}{4r^2}} \int_{h_{\min}}^{h_{\max}} e^{-\frac{1}{r^2}\left(h^2 - 2\frac{\boldsymbol{x}^{(n)}+\boldsymbol{x}^{(n')}}{2}h + \left(\frac{\boldsymbol{x}^{(n)}+\boldsymbol{x}^{(n')}}{2}\right)^2\right)} dh$$

$$= \sqrt{\pi r^2} S \exp\left\{-\frac{(\boldsymbol{x}^{(n)} - \boldsymbol{x}^{(n')})^2}{4r^2}\right\} = C(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n')})$$

The prior can be summarized by covariance function $C(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n')})$

Gaussian process

Conclusion: The prior probability for $\boldsymbol{t}$ can be written as
$P(\boldsymbol{t}) = \frac{1}{Z} e^{-\frac{1}{2}\boldsymbol{t}^T \boldsymbol{C}^{-1}\boldsymbol{t}}$, where $C_{nn'} = C(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n')}) + \sigma_\nu^2 \delta_{nn'}$.
We don't need to construct the RBF estimator.

Inference of t_{N+1} can now be made by computing
$P(t_{N+1}|\boldsymbol{t}_N) = \frac{P(t_{N+1},\boldsymbol{t}_N)}{P(\boldsymbol{t}_N)} \propto \exp\left\{-\frac{1}{2}\left[\boldsymbol{t}_N\ t_{N+1}\right]^T \boldsymbol{C}_{N+1}^{-1}\ \left[\boldsymbol{t}_N\ t_{N+1}\right]\right\}$

We can evaluate the mean and stdev of the posterior of t_{N+1} by brute-force inversion of $\boldsymbol{C}_{N+1}$.

GP: Computational complexity

Matrix inversion scales as $O(N^3)$. The computational complexity can be reduced a bit by partitioned inverse equations. Define submatrices

$$C_{N+1} = \left[\begin{array}{cc} \left[\quad \boldsymbol{C}_N \quad \right] & \left[\boldsymbol{k} \right] \\ \left[\quad \boldsymbol{k}^T \quad \right] & [\kappa\ ] \end{array} \right].$$

Using the submatrices, it turns out that

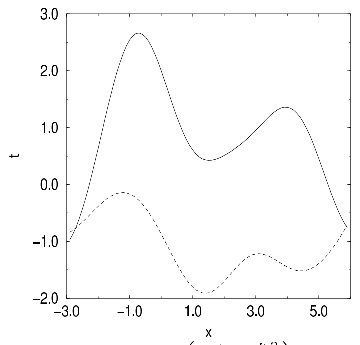
$$p(t_{N+1}|\boldsymbol{t}_N) = \frac{1}{Z} \exp\left\{ -\frac{(t_{N+1}-\hat{t}_{N+1})^2}{2\hat{\sigma}^2_{\hat{t}_{N+1}}} \right\},$$

where $\hat{t}_{N+1} = \boldsymbol{k}^T \boldsymbol{C}_N^{-1} \boldsymbol{t}_N$, and $\hat{\sigma}^2_{\hat{t}_{N+1}} = \kappa - \boldsymbol{k}^T \boldsymbol{C}_N^{-1} \boldsymbol{k}$.

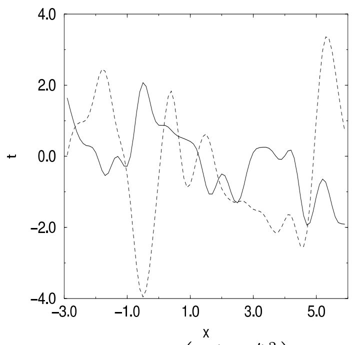
$\Rightarrow$ *We only need to invert $\boldsymbol{C}_N$.*

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

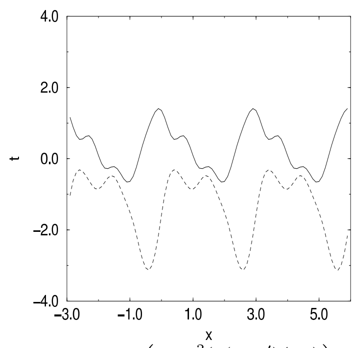
C determines the behavior of the GP. The restrictions to C are that it must be positive definite and symmetric.



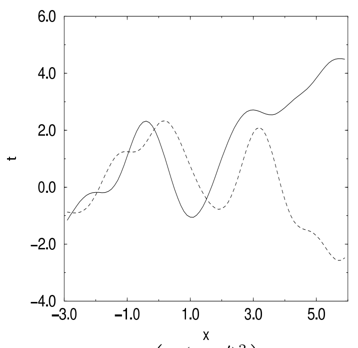
(a) $2\exp\left(-\frac{(x-x')^2}{2(1.5)^2}\right)$



(b) $2\exp\left(-\frac{(x-x')^2}{2(0.35)^2}\right)$



(c) $2\exp\left(-\frac{\sin^2(\pi(x-x')/3.0)}{2(0.5)^2}\right)$



(d) $2\exp\left(-\frac{(x-x')^2}{2(1.5)^2}\right) + xx'$

Preferred functional forms

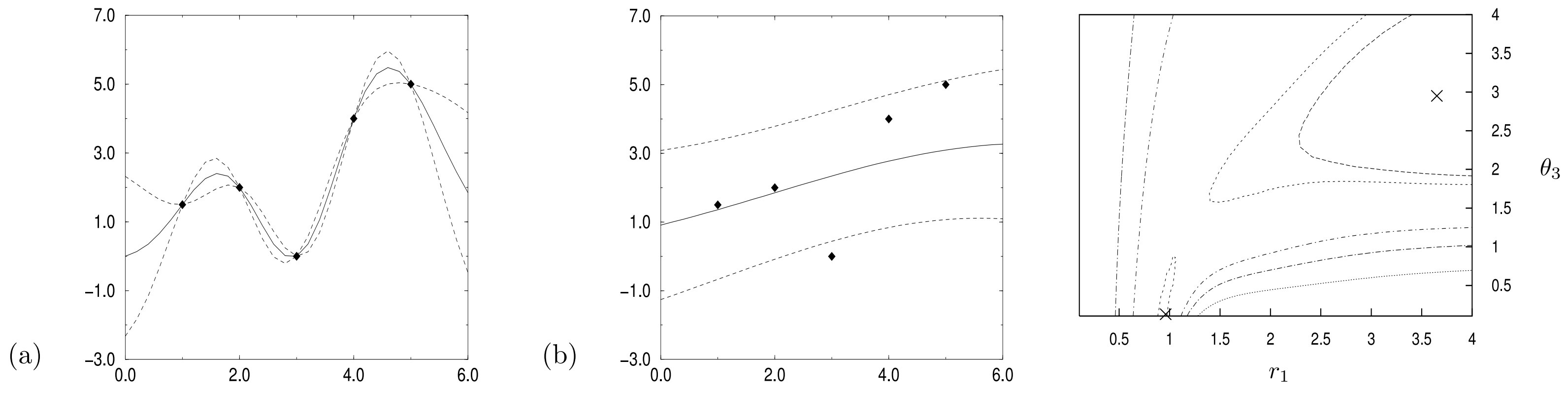
The prior gives high probability to vectors which have small $\boldsymbol{t}^T \boldsymbol{C}^{-1} \boldsymbol{t}$. This is in particular the case for the normalized eigenvectors v_i of $\boldsymbol{C}$ with large eigenvalues λ_i, since
$\boldsymbol{C} v_i = \lambda_i v_i$ yields $v_i^T \boldsymbol{C}^{-1} v_i = \lambda_i^{-1}$.

The prior is thus biased towards solutions with small λ_i^{-1}.

We can therefore view the preferred functional forms by plotting the eigenvectors. The associated eigenvalues tell the degree of preference of the functional form.

On Adaptation

The covariance function depends on hyperparameters Θ. Optimal values for Θ can be found by using a MAP estimate or by MCMC.
Remember: The posterior may be multimodal.



Summary

- GP places an explicit Gaussian prior on the form of the functions.
- GP is defined by mean and covariance *functions*.
- Computationally heavy, $\mathcal{O}(N^3)$
- Classification task is more difficult than regression, since the likelihood function $P(\boldsymbol{t}_N|y(\boldsymbol{x}), \boldsymbol{X}_N)$ is not Gaussian. We either have to resort to (Laplace) approximations, MAP, or MCMC.
- $\boldsymbol{C}$ can be seen as a kernel $\Rightarrow$ connection to Reproducing Kernel Hilbert Spaces.

Exercise 1

16.6 (inference and variance) For a normal distribution in two variables with

$$C = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 0.75 \end{bmatrix}$$

as covariance and zero mean, compute the variance in terms of the first variable if the second one is observed, and vice versa.

From Schölkopf & Smola: Learning with Kernels

Exercise 2

16.7 (Samples from a Gaussian Process prior) Draw a sample $\boldsymbol{X}$ at random from the uniform distribution on $[0,1]^2$ and compute the corresponding covariance matrix $\boldsymbol{C}$. Use for instance the linear covariance function $C(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{x}\boldsymbol{x}'$ and the Gaussian RBF covariance function $C(\boldsymbol{x},\boldsymbol{x}') = \exp\left\{-\frac{1}{2\sigma^2}\|\boldsymbol{x}-\boldsymbol{x}'\|^2\right\}$.
Write a program which draws samples uniformly from the normal distribution $\mathcal{N}(0,\boldsymbol{C})$ (Hint: Compute the eigenvectors of $\boldsymbol{C}$ first). What difference do you observe when using different covariance functions?

From Schölkopf & Smola: Learning with Kernels