To appear in: The Handbook of Brain Theory and Neural Networks, Second edition, (M.A. Arbib, Ed.), Cambridge, MA: The MIT Press, 2002. http://mitpress.mit.edu, ©The MIT Press.

Gaussian Processes[1]

C. K. I. Williams

Institute for Adaptive and Neural Computation

Division of Informatics, University of Edinburgh

5 Forrest Hill, Edinburgh EH1 2QL, Scotland, UK

Tel: +44 131 651 1212

fax: +44 131 650 6899

Email: c.k.i.williams@ed.ac.uk

Web: http://anc.ed.ac.uk

March 1, 2002

[1]This paper was prepared within the space and reference limitations of HBTNN2e. I thank David Sterratt, Michael Arbib and an anonymous referee for comments that helped improve the paper.

INTRODUCTION

Much of the work in the field of artificial neural networks concerns the problem of supervised learning. Here we may be interested in regression problems (by which we mean the prediction of some real-valued variable(s)), or classification problems (predicting a class label) given the values of some input variables. Due to factors such as measurement noise, it is necessary to take a statistical view of the learning problem.

Given (possibly noisy) observations of a function at n points, it is necessary to impose extra assumptions about the function if there is to be hope of predicting its value elsewhere. Here we take a Bayesian approach, placing a prior probability distribution over possible functions and then letting the observed data "sculpt" this prior into a posterior using the available data. The Bayesian approach can provide solutions to several problems such as local optima in weight space, the setting of regularization parameters, overfitting and model selection (see MacKay 1992, Neal 1996 and BAYESIAN METHODS FOR SUPERVISED NEURAL NETWORKS).

One can place a prior distribution $P(\mathbf{w})$ on the weights $\mathbf{w}$ of a neural network to induce a prior over functions $P(y(\mathbf{x};\mathbf{w}))$ but the computations required to make predictions are not easy due to the non-linearities in the system, and one needs to resort to analytic approximations or Monte Carlo methods. Gaussian processes are a way of specifying a prior directly over function space; it is often simpler to do this than to work with priors over parameters. Gaussian processes (GPs) are probably the simplest kind of function space prior that one can consider, being a generalization of finite-dimensional Gaussian distributions over vectors.

A finite-dimensional Gaussian distribution is defined by a mean vector and covariance matrix. A GP is defined by a mean function (which we shall usually take to be identically zero), and a *covariance function* $C(\mathbf{x}, \mathbf{x}')$, which indicates how correlated the value of the function y is at $\mathbf{x}$ and $\mathbf{x}'$. This function encodes our assumptions about the problem (for example that the function is smooth and continuous) and will influence the quality of the predictions.

Gaussian process prediction is illustrated in Figure 1. The upper panel shows a sample of 5 functions drawn from the prior. The lower panel shows 5 samples from the posterior after two observations have been made; notice that the posterior is tightly constrained near to the observations, but varies more widely further away. Essentially what has happened is that prior samples not consistent with the observations have been eliminated. The crucial computational point is that it is not necessary to draw samples to make predictions; for regression problems only linear algebra is required. Below we give more detail on this computation, discuss how to use GPs for classification problems, and describe how data can be used to adapt the covariance function to the given prediction problem.

Further discussion of Gaussian processes is available in Schölkopf and Smola (2001), MacKay (1998) and Williams (1998).

GAUSSIAN PROCESSES

Formal definition

A stochastic process is a collection of random variables $\{Y(\mathbf{x}) | \mathbf{x} \in X\}$ indexed by a set X. In our case X will often be $\mathbb{R}^d$, where d is the number of inputs. The stochastic process is specified by giving the joint probability distribution for every finite subset of variables $Y(\mathbf{x}_1), \ldots, Y(\mathbf{x}_k)$ in a consistent manner. A Gaussian process (GP) is a stochastic process for which any finite set of Y-variables has a joint multivariate Gaussian distribution. A GP is fully specified by its mean function $\mu(\mathbf{x}) = E[Y(\mathbf{x})]$ and its covariance function $C(\mathbf{x}, \mathbf{x}') = E[(Y(\mathbf{x}) - \mu(\mathbf{x}))(Y(\mathbf{x}') - \mu(\mathbf{x}'))]$. For a multidimensional input space a Gaussian process may also be called a Gaussian random field.

Below we consider Gaussian processes which have $\mu(\mathbf{x}) \equiv 0$. A non-zero $\mu(\mathbf{x})$ can be incorporated into the framework at the expense of a little extra complexity.

Example: Bayesian linear regression

Consider the model $y(x) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$, where $\{\phi_i\}$ is a set of fixed basis functions and $\mathbf{w}$ is a vector of "weights". Let $\mathbf{w}$ have a Gaussian distribution with mean $\mathbf{0}$ and covariance Σ. Then $\mu(\mathbf{x}) = E[y(\mathbf{x})] = E[\mathbf{w}^T]\boldsymbol{\phi}(\mathbf{x}) = 0$ as $E[\mathbf{w}] = \mathbf{0}$. As the mean is zero we have that $C(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}^T(\mathbf{x}) E[\mathbf{w}\mathbf{w}^T] \boldsymbol{\phi}(\mathbf{x}') = \boldsymbol{\phi}^T(\mathbf{x}) \Sigma \boldsymbol{\phi}(\mathbf{x}')$. For example, using basis functions 1 and the components of $\mathbf{x}$ along with $\Sigma = I$ gives $C(\mathbf{x}, \mathbf{x}') = 1 + \mathbf{x}.\mathbf{x}'$.

In the case of a finite dimensional model we can make predictions using calculations in the parameter space (of dimension m), or a GP prediction (which is n dimensional, where n is the number of data points). For $m < n$ the parameter space method is preferable, but for many useful covariance functions (see, e.g. equation 1) m is infinite and the GP method is necessary.

Covariance functions

The only constraint on the covariance function is that it should generate a non-negative definite covariance matrix for any set of points in X. This gives wide scope, and different choices of $C(\mathbf{x}, \mathbf{x}')$ can give rise to such differing priors as straight lines of the form $y = w_0 + w_1 x$ (as discussed above) to the very rough and jagged sample paths associated with a Wiener process (a model for Brownian motion) or an Ornstein-Uhlenbeck process.

One very common form of covariance function is the *stationary* covariance function, where $C(\mathbf{x}, \mathbf{x}')$ is a function of $\mathbf{x} - \mathbf{x}'$. The use of stationary covariance functions is appealing if one would like the predictions to be invariant under shifts of the origin in input space. For example, in one dimension letting $h = x - x'$, the covariance of the Ornstein-Uhlenbeck process is $C_{OU}(h) = v_0 e^{-|h|/\lambda}$, where v_0 sets the overall variance of the process and λ sets a lengthscale in the input space. Another example of a stationary covariance function is the "squared exponential" covariance function $C_{SE}(h) = v_0 \exp(-h^2/\lambda^2)$ (sometimes called the "Gaussian" covariance function).

One commonly-used covariance function for inputs in $\mathbb{R}^d$ is

$$C(\mathbf{x}, \mathbf{x}') = v_0 \exp\{-\sum_{l=1}^{d} \frac{(x_l - x_l')^2}{\lambda_l^2}\}. \tag{1}$$

This is simply the product of d squared-exponential covariance functions, but with different lengthscales on each dimension. The general form of the covariance function expresses the idea that cases with nearby inputs will have highly correlated outputs, and the λ parameters allow a different distance measure for each input dimension. For irrelevant inputs, the corresponding λ_l will become large, and the model will effectively ignore that input. This is closely related to the Automatic Relevance Determination (ARD) idea of

MacKay and Neal (Neal, 1996).

The term kernel function used in the Support Vector Machines literature is broadly equivalent to the covariance function. Further information on kernel/covariance functions can be found in Schölkopf and Smola (2001, chapters 4 and 13), MacKay (1998), Williams (1998) and references therein.

GAUSSIAN PROCESSES FOR REGRESSION PROBLEMS

Above we have discussed the properties of Gaussian processes. We now assume that we have input points $\mathbf{x}^n = \mathbf{x}_1, \ldots, \mathbf{x}_n$ and target values $\mathbf{t} = t_1, \ldots, t_n$, and wish to predict the function value y_* corresponding to an input $\mathbf{x}_*$. We assume that the target values t_i are obtained from the corresponding function value y_i by means of additive Gaussian noise, i.e. $t_i = y_i + \epsilon_i$ for $i = 1, \ldots, n$, where ϵ_i is an independent zero-mean Gaussian random variable of variance σ_ν^2. (The generalization to different variances at each location is straightforward, but notationally a bit more complex.) As the prior is a Gaussian process, the prior distribution over the y_i's is given by $\mathbf{Y} \sim N(\mathbf{0}, K)$, where K is the $n \times n$ covariance matrix with entries $K_{ij} = C(\mathbf{x}_i, \mathbf{x}_j)$. It is then easy to show that the prior distribution over the targets is $N(\mathbf{0}, K + \sigma_\nu^2 I_n)$ where I_n is the $n \times n$ identity matrix.

To make a prediction for y_* we now need to consider the $n+1$-dimensional vector which consists of the n variables in $\mathbf{t}$ with the variable y_* appended, and condition on $\mathbf{t}$ to obtain $P(y_*|\mathbf{t})$. As conditional distributions of jointly Gaussian variables are also Gaussian, it is clear that this distribution will be

Gaussian, and our task is to compute the mean $\hat{y}(\mathbf{x}_*)$ and variance $\hat{\sigma}^2(\mathbf{x}_*)$. It turns out that

$$\begin{aligned} \hat{y}(\mathbf{x}_*) &= \mathbf{k}^T(\mathbf{x}_*)(K + \sigma_\nu^2 I_n)^{-1}\mathbf{t} = \sum_{i=1}^{n} \alpha_i C(\mathbf{x}_i, \mathbf{x}_*), && (2) \\ \hat{\sigma}^2(\mathbf{x}_*) &= C(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}^T(\mathbf{x}_*)(K + \sigma_\nu^2 I_n)^{-1}\mathbf{k}(\mathbf{x}_*), && (3) \end{aligned}$$

where $\mathbf{k}(\mathbf{x}_*)$ is the $n \times 1$ vector of covariances $(C(\mathbf{x}_1, \mathbf{x}_*), \ldots, C(\mathbf{x}_n, \mathbf{x}_*))^T$, and $\boldsymbol{\alpha} = (K + \sigma_\nu^2 I_n)^{-1}\mathbf{t}$. Unpacking equation (2), we see that the prediction function $\hat{y}(\mathbf{x}_*)$ is a linear combination of the kernel functions $C(\mathbf{x}_i, \mathbf{x}_*)$, with coefficients given by the appropriate entries of the vector $\boldsymbol{\alpha}$.

Equations 2 and 3 require the inversion of a $n \times n$ matrix which is in general a $O(n^3)$ operation. When n is of the order of a few hundred then this is quite feasible with modern computers. However, once $n \sim O(1000)$ these computations can be quite time consuming, and much recent research effort has gone into developing approximation methods; see Tresp (2001) for a review. Note that in special cases (notably when the input space is $\mathbb{R}$ and for certain Markovian kernels), the necessary calculations can be carried out in linear time (see, Wahba, 1990 for further details).

The use of Gaussian processes for regression problems has been studied extensively by Carl Rasmussen in (Rasmussen, 1996) and in his PhD thesis (available from `http://www.cs.utoronto.ca/~carl/`). He carried out a careful comparison of the Bayesian treatment of Gaussian process regression with several other state-of-the-art methods on a number of problems and found that its performance is comparable to that of Bayesian neural networks as developed by Neal (1996), and consistently better than the other methods tested.

Adapting the covariance function

Given a covariance function it is straightforward to make predictions for new test points. However, in practical situations we are unlikely to know which covariance function to use. One option is to choose a parametric family of covariance functions (with a parameter vector $\boldsymbol{\theta}$) and then to search for parameters that give good predictions.

Adaptation of $\boldsymbol{\theta}$ is facilitated by the fact that the log likelihood $l = \log P(\mathbf{t}|\boldsymbol{\theta})$ can be calculated analytically as

$$l = -\frac{1}{2}\log\det(K + \sigma_\nu^2 I_n) - \frac{1}{2}\mathbf{t}^T(K + \sigma_\nu^2 I_n)^{-1}\mathbf{t} - \frac{n}{2}\log 2\pi. \tag{4}$$

This is just the log likelihood of the vector $\mathbf{t}$ under a Gaussian with mean $\mathbf{0}$ and covariance $K + \sigma_\nu^2 I_n$. The evaluation of the likelihood and its partial derivatives with respect to the parameters takes time $O(n^3)$, unless special structure in the problem can be exploited. Given l and its derivatives with respect to $\boldsymbol{\theta}$ it is straightforward to feed this information to an optimization package in order to obtain a local maximum of the likelihood.

One can also combine $P(\mathbf{t}|\boldsymbol{\theta})$ with a prior $P(\boldsymbol{\theta})$ to yield a Bayesian approach to the problem. Another approach to adapting $\boldsymbol{\theta}$ is to use the cross-validation (CV) or generalized cross-validation (GCV) methods, as discussed in Wahba (1990).

Relationship to other methods

Prediction with Gaussian processes is certainly not a very recent topic; the basic theory goes back at least as far as the work of Wiener and Kolmogorov in the 1940's on time series. Gaussian process prediction is also well known in the geostatistics field (see Cressie, 1993) where it is known as "kriging", although this literature naturally has focussed mostly on two- and three-dimensional input spaces.

As mentioned above, there is a close relationship between Bayesian approaches and regularization theory. This connection was described in Kimeldorf and Wahba (1970), and further details can be found in Wahba (1990), Poggio and Girosi (1990) and GENERALIZATION AND REGULARIZATION IN NONLINEAR LEARNING SYSTEMS (q.v.).

When the covariance function $C(\mathbf{x}, \mathbf{x}')$ depends only on $h = |\mathbf{x} - \mathbf{x}'|$, the predictor derived in equation 2 has the form $\sum_i c_i C(|\mathbf{x} - \mathbf{x}_i|)$ and may be called a *Radial Basis Function* (or RBF) network. This is one derivation of RBFs, which are described in more detail in RADIAL BASIS FUNCTION NETWORKS.

The Gaussian process approach adds a stochastic process view to the regularization viewpoint, giving us "error bars" on the prediction (equation 3), an expression for $P(\mathbf{t}|\boldsymbol{\theta})$ and its derivatives, and allows us to use the Bayesian machinery for hierarchical models.

GAUSSIAN PROCESSES FOR CLASSIFICATION PROBLEMS

Given training data and an input $\mathbf{x}$, the aim of a classifier is to predict the corresponding class label. This may be done by simply predicting a class label ("hard" classification), or by outputting an estimate of the posterior probabilities for each class $P(k|\mathbf{x})$ ("soft" classification), where $k = 1, \ldots C$

indexes the C classes. Naturally we require that $0 \leq P(k|\mathbf{x}) \leq 1$ for all k and that $\sum_k P(k|\mathbf{x}) = 1$. A naïve application of the regression method for Gaussian processes using, say, targets of 1 when an example of class k is observed and 0 otherwise will not obey these constraints. Soft classification has the advantage that the posterior probability estimates can be used in a principled fashion with loss matrices, rejection thresholds *etc.*

For the two-class classification problem it is only necessary to represent $P(1|\mathbf{x})$, since $P(2|\mathbf{x}) = 1 - P(1|\mathbf{x})$. An easy way to ensure that the estimate $\pi(\mathbf{x})$ of $P(1|\mathbf{x})$ lies in $[0, 1]$ is to obtain it by passing an unbounded value $y(\mathbf{x})$ through an appropriate function which has range $[0, 1]$. A common choice is the logistic function $\sigma(z) = 1/(1 + e^{-z})$ so that $\pi(\mathbf{x}) = \sigma(y(\mathbf{x}))$. The input $y(\mathbf{x})$ to the logistic function will be called the *activation*. In the simplest method of this kind, logistic regression, the activation is simply computed as a linear combination of the inputs, plus a bias, i.e. $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$. Using a Gaussian process or other flexible methods allow $y(\mathbf{x})$ to be a non-linear function of the inputs.

For the classification problem with more than two classes, a simple extension of this idea using the "softmax" function gives the predicted probability for class k as

$$\pi(k|\mathbf{x}) = \frac{\exp y_k(\mathbf{x})}{\sum_m \exp y_m(\mathbf{x})}. \tag{5}$$

For the rest of this section we shall concentrate on the two-class problem; extension of the methods to the multi-class case is relatively straightforward.

Defining a Gaussian process prior over the activation $y(\mathbf{x})$ automatically induces a prior over $\pi(\mathbf{x})$. To make predictions for a test input $\mathbf{x}_*$ when using fixed parameters in the GP we would like to compute $\hat{\pi}_* = \int \pi_* P(\pi_*|\mathbf{t}, \boldsymbol{\theta}) \, d\pi_*$, which requires us to find $P(\pi_*|\mathbf{t}) = P(\pi(\mathbf{x}_*)|\mathbf{t})$ for a new input $\mathbf{x}_*$. This can be done by finding the distribution $P(y_*|\mathbf{t})$ (y_* is the activation of π_*) as given by

$$P(y_*|\mathbf{t}) = \int P(y_*|\mathbf{y})P(\mathbf{y}|\mathbf{t})d\mathbf{y} = \frac{1}{P(\mathbf{t})}\int P(y_*|\mathbf{y})P(\mathbf{y})P(\mathbf{t}|\mathbf{y})d\mathbf{y}, \quad (6)$$

where $\mathbf{y} = (y_1, \ldots, y_n)$ denotes the activations corresponding to the data-points. $P(\pi_*|\mathbf{t})$ can then be found from $P(y_*|\mathbf{t})$ using the appropriate Jacobian to transform the distribution. When $P(\mathbf{t}|\mathbf{y})$ is Gaussian then the integral in equation 6 can be computed exactly to yield equations 2 and 3. However, the usual expression for $P(\mathbf{t}|\mathbf{y}) = \prod_i P(t_i|y_i)$ and $P(t_i|y_i) = \pi_i$ if $t_i = 1$ and $P(t_i|y_i) = (1 - \pi_i)$ for $t_i = -1$ for classification data (where the t's take on values of 1 or -1), means that the marginalization to obtain $P(y_*|\mathbf{t})$ is no longer analytically tractable. Faced with this problem we can either use an analytic approximation to the integral in equation 6 or use Monte Carlo methods to approximate it. These two approaches will be considered in turn.

First we note that $P(y_*|\mathbf{t})$ is mediated through $P(\mathbf{y}|\mathbf{t})$ and that $P(y_*|\mathbf{y})$ is Gaussian, so that obtaining information about $P(\mathbf{y}|\mathbf{t})$ is the essential step. It is easy to find the maximum of this distribution by optimizing $\log P(\mathbf{y}) + \log P(\mathbf{t}|\mathbf{y})$ with respect to $\mathbf{y}$, e.g. with a Newton-Raphson iteration. It can be shown that the optimization problem is convex. This yields the *maximum a posteriori* estimator $\mathbf{y}^{MAP}$. We could build a classifier based on $\mathbf{y}^{MAP}$ by calculating $y^{MAP}(\mathbf{x}_*)$ as the mean of $P(y_*|\mathbf{y}^{MAP})$. This can then be fed through the logistic function to obtain an approximation to $\hat{\pi}_*$. This MAP solution is the one used in spline smoothing approaches to classification (Wahba, 1990).

One can also make a Gaussian approximation to $P(\mathbf{y}|\mathbf{t})$ with mean $\mathbf{y}^{MAP}$

and inverse covariance matrix $-\nabla\nabla \log P(\mathbf{y}|\mathbf{t})$. This yields a Laplace approximation to the integral in equation 6.

Neal (1998) has developed a MCMC method for the Gaussian process classification model. This works by generating samples from $P(\mathbf{y}|\mathbf{t})$ by updating each of the n individual y_i's sequentially using Gibbs sampling. This sampling process can be also be interleaved with sampling for the parameters $\boldsymbol{\theta}$. As with the regression problem, there has been much work on approximation schemes for large data sets, see Tresp (2001) for further details.

Classifiers using splines have been used extensively on a wide variety of problems, see Wahba (1990) and references in GENERALIZATION AND REGULARIZATION IN NONLINEAR LEARNING SYSTEMS. Gaussian process classifiers using MCMC sampling over $\boldsymbol{\theta}$ have been described in Williams and Barber (1998).

Relationship to Support Vector Machines

Above we have seen that the *maximum a posteriori* solution $\mathbf{y}^{MAP}$ is obtained by minimizing $\Psi(\mathbf{y}) = -\log P(\mathbf{y}) - \log P(\mathbf{t}|\mathbf{y})$. This expression can be refined using $-\log P(\mathbf{t}|\mathbf{y}) = -\sum_i \log P(t_i|y_i)$ and $-\log P(t_i|y_i) = \log(1 + e^{-t_i y_i})$ to give

$$\Psi(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T K^{-1}\mathbf{y} + \sum_i \log(1 + e^{-t_i y_i}) + c \tag{7}$$

where c is a constant independent of $\mathbf{y}$. The criterion optimized by the Support Vector Machine (SVM) learning algorithm (Vapnik, 1995) is very similar, but with $g_{GP}(z) \stackrel{def}{=} \log(1+e^{-z})$ replaced by $g_{SVM}(z) \stackrel{def}{=} [1-z]_+$, where $[x]_+ = \max(x, 0)$. These are both monotonically decreasing functions of z which are linear for $z \to -\infty$. They both decay to zero as $z \to \infty$, but the main difference is that the g_{SVM} takes on the value 0 for $z > 1$, while g_{GP} asymptotes to 0 as $z \to \infty$. The SVM optimization problem is convex, but inequality constraints mean that it is quadratic programming problem.

By replacing g_{GP} with g_{SVM} we obtain $y^{SVM}(\mathbf{x}_*)$ instead of $y^{MAP}(\mathbf{x}_*)$. To make a "hard" $(+1/-1)$ prediction we simply take the predicted class label as $\mathrm{sgn}(y(\mathbf{x}_*))$. This is the SVM classifier. The effect of the flat region of g_{SVM} is to introduce *sparsity* into the prediction of the corresponding $y^{SVM}(\mathbf{x}_*)$, where only those data points with $t_i y_i \leq 1$ contributing; these are known as the support patterns. Note that g_{SVM} is not interpretable as a negative log likelihood as it does not normalize properly. For further discussion see Wahba (1999) and SUPPORT VECTOR MACHINES (q.v.).

DISCUSSION

In this article we have seen how Gaussian process priors over functions (which are in general infinite-dimensional objects) can be used in a computationally efficient manner to make predictions.

Methods such as Gaussian Processes and Support Vector Machines have come to be known under the umbrella term of *kernel machines*, see (Schölkopf and Smola, 2001). The website `http://www.kernel-machines.org/` has extensive links to research publications and software in this area.

One key issue concerning obtaining good performance with kernel methods is the choice of kernel. The squared-exponential kernel is widely used in practice, but it only encodes a general notion of smoothness. For particular problems incorporation of prior/domain knowledge requires "kernel engineering". A second key issue for kernel methods is developing good approximation algorithms for large datasets.

References

Cressie, N. A. C. (1993). Statistics for Spatial Data. Wiley, New York.

Kimeldorf, G. and Wahba, G. (1970). A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. Annals of Mathematical Statistics, 41:495–502.

MacKay, D. J. C. (1992). A Practical Bayesian Framework for Backpropagation Networks. Neural Computation, 4(3):448–472.

* MacKay, D. J. C. (1998). Introduction to Gaussian Processes. In Bishop, C. M., editor, Neural Networks and Machine Learning. Springer-Verlag.

Neal, R. M. (1998). Regression and classification using Gaussian process priors (with discussion). In Bernardo, J. M. *et al*, eds Bayesian Statistics 6. Oxford University Press, pp. 475-501.

Neal, R. M. (1996). Bayesian Learning for Neural Networks. Springer, New York. Lecture Notes in Statistics 118.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. Proceedings of IEEE, 78:1481–1497.

Rasmussen, C. E. (1996). A Practical Monte Carlo Implementation of Bayesian Learning. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, Advances in Neural Information Processing Systems 8. MIT Press, pp. 598-604.

* Schölkopf, B. and Smola. A. (2001), Learning with Kernels. MIT Press. (Gaussian processes are covered in chapter 16.)

Tresp, V. (2001). Scaling Kernel-Based Systems to Large Data Sets. Data Mining and Knowledge Discovery, 5(3):197-211.

Vapnik, V. N. (1995). The Nature of Statistical Learning Theory. Springer Verlag, New York.

* Wahba, G. (1990). Spline Models for Observational Data. SIAM. CBMS-NSF Regional Conference Series in Applied Mathematics. (Chapter 1 provides a good overview.)

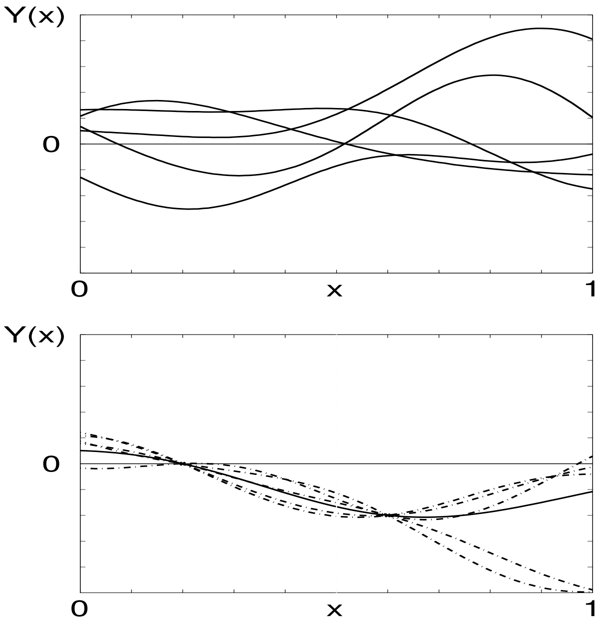
Wahba, G. (1999). Support Vector Machines, Reproducing Kernel Hilbert Spaces, and Randomized GACV. In Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, Advances in Kernel Methods. MIT Press, pp. 69-88.

* Williams, C. K. I. (1998). Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In Jordan, M. I., editor, Learning in Graphical Models, pages 599–621. Kluwer Academic.

Williams, C. K. I. and Barber, D. (1998). Bayesian Classification with Gaussian Processes. IEEE Trans. Pattern Analysis and Machine Intelligence, 20(12):1342–1351.

FIGURE CAPTION

Figure 1. Top: Five samples from a Gaussian process prior. Bottom: Five samples from the Gaussian process posterior (shown as dot-dash lines) and the posterior mean (solid line), after observing the data points $(0.2, 0)$ and $(0.6, -1)$.


Y(x)
0
0
x
1
Y(x)
0
0
x
1