Regression and Classification Using Gaussian Process Priors

Radford M. Neal

Departments of Statistics and Computer Science
University of Toronto

http://www.cs.utoronto.ca/~radford/
radford@stat.utoronto.ca

6th Valencia Meeting, 1998

Outline of Talk

1) Bayesian modeling with Gaussian processes.
   - Regression models, linear and beyond.
   - Classification models, using latent values.
   - Varieties of covariance functions, and their hyperparameters.

2) Computations for Gaussian process models.
   - Making predictions.
   - Sampling for latent values.
   - Sampling for hyperparameters.

3) Example: A three-way classification task.

4) History of Gaussian process models.
   - Past usage of Gaussian process models.
   - Recent work in the neural networks community.
   - Why aren't Gaussian process models a standard tool for Bayesian modeling?

Regression and Classification Problems

We observe n cases, numbered $i = 1, \ldots, n$. For each case, we know a vector of *inputs* (predictors, covariates), $x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}$ and a *target* (response), $t^{(i)}$.

For a new case with known inputs, $x^{(n+1)}$, we wish to predict the unknown target, $t^{(n+1)}$.

We can do this by modeling the conditional distribution of t given x, based on some function $f(x)$ found from the data.

For regression, the targets are real-valued. In a Gaussian model,

$$t \mid x \;\sim\; N(f(x), \sigma^2)$$

For classification, the targets are categorical. For a binary logistic model,

$$t \mid x \;\sim\; \text{Bernoulli}\left([1 + \exp(-f(x))]^{-1}\right)$$

Gaussian Process Priors

We can give the function $f(x)$ in our model a Gaussian process prior.

A Gaussian process is a distribution over functions $f(x)$ for which the joint distribution of $f(x^{(1)}), \ldots, f(x^{(n)})$ at any $x^{(1)}, \ldots, x^{(n)}$ is multivariate Gaussian.

We specify the Gaussian process by giving:

- The mean of $f(x^{(i)})$ as a function of $x^{(i)}$.
- The covariance of $f(x^{(i)})$ and $f(x^{(j)})$ as a function of $x^{(i)}$ and $x^{(j)}$.

Note: It will be convenient to also refer to "Gaussian processes" over "functions" that have noise — ie, for which the values at $x^{(i)}$ and $x^{(j)}$ can differ even when $x^{(i)} = x^{(j)}$.

Bayesian Linear Regression as a Gaussian Process Model

Consider a linear regression model:

$$t^{(i)} \;=\; \alpha \;+\; \sum_{u=1}^{p} x_u^{(i)} \beta_u \;+\; \epsilon^{(i)}$$

where $\epsilon^{(i)}$ is Gaussian noise with variance σ_ϵ^2.

We give the unknown parameters α and the β_u independent Gaussian priors with means of 0 and with variances of σ_α^2 and σ_u^2 $(u = 1, \ldots, p)$.

We then find that $E[t^{(i)}] = 0$, and that

$$\text{Cov}[t^{(i)}, t^{(j)}] \;=\; \sigma_\alpha^2 \;+\; \sum_{u=1}^{p} x_u^{(i)} x_u^{(j)} \sigma_u^2 \;+\; \delta_{ij} \sigma_\epsilon^2$$

This model can thus be seen as a Gaussian process, specified by giving the covariances of targets for different inputs.

Exponential Covariance Functions

We can get non-linear functions of a univariate x using covariance functions of the form

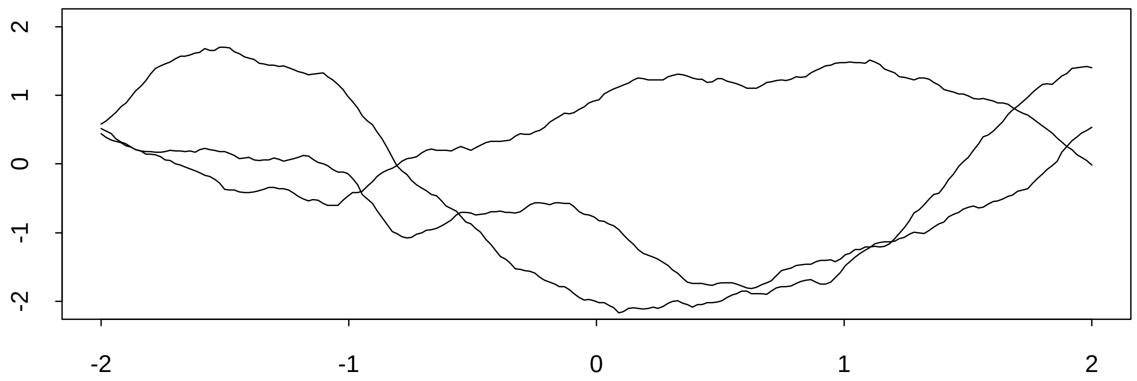
$$\text{Cov}\Big[t^{(i)}, t^{(j)}\Big] \;=\; \eta^2 \exp\Big(-\rho^R \Big|x^{(i)} - x^{(j)}\Big|^R\Big)$$

where $0 < R \leq 2$.

If we let the mean be zero, this yields a stationary process, with variance η^2, and with a length scale fixed by ρ.

If $R = 2$, functions drawn from this process are differentiable; if $R < 2$, they are not.

Sample realizations with $R = 1.8$, $\eta = 1$, $\rho = 1$:



Classification Models Based on Gaussian Processes

We can't build a classification model directly as a Gaussian Process. But we can express the class probabilities in terms of "latent values", which we do model as a Gaussian process.

For a binary classification model, we can use latent values $y^{(i)}$, and let

$$P(t^{(i)} = 1) \ = \ [1 + \exp(-y^{(i)})]^{-1}$$

For a K-way classification model, we can use K latent values, $y_0^{(i)}, \ldots, y_{K-1}^{(i)}$, and let

$$P(t^{(i)} = k) \ = \ \exp(-y_k^{(i)}) \ \Big/ \ \sum_{k'=0}^{K-1} \exp(-y_{k'}^{(i)})$$

The K latent value functions can be given independent Gaussian process priors.

"Jitter" in Classification Models

In practice, it is usually necessary to include at least a small amount of "jitter" in the covariance function for latent values.

For example, we might use:

$$\text{Cov}\left[y^{(i)}, y^{(j)}\right] \;=\; \exp\left(-\left(x^{(i)}-x^{(j)}\right)^2\right) + 0.1^2\delta_{ij}$$

The jitter of 0.1 acts like added noise.

A small amount of jitter makes computations easier, while having little effect on the model.

A large amount of jitter turns the logit model into a probit model, since the logistic function looks like a threshold function when latent values are typically large.

New Covariance Functions From Old

We can build new covariance functions from old ones by taking sums and products.

A sum of covariance functions produces an additive model, with independent priors for the components. These components can be

- Functions of different inputs, or subsets of the inputs.
- Functions with different scales of variation.
- Functions with covariance functions of different types.

A product of univariate covariance functions can be used to define a covariance function for a non-additive model in which the inputs interact.

Covariance Functions with Unknown Hyperparameters

Here is one generally useful multivariate covariance function defined using products:

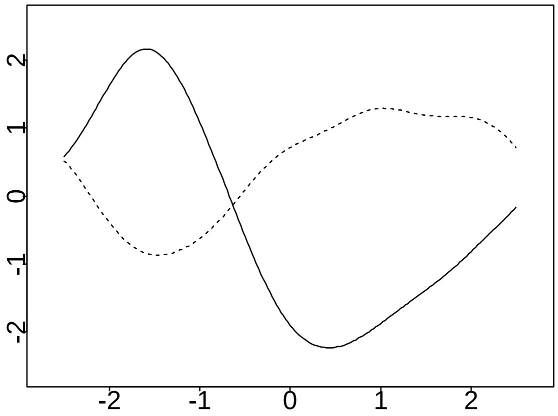
$$\begin{aligned}
\text{Cov}\Big[t^{(i)}, t^{(j)}\Big] &= \eta^2 \prod_{u=1}^{p} \exp\Big(-\rho_u^R \Big|x_u^{(i)} - x_u^{(j)}\Big|^R\Big) \\
&= \eta^2 \exp\Big(-\sum_{u=1}^{p} \rho_u^R \Big|x_u^{(i)} - x_u^{(j)}\Big|^R\Big)
\end{aligned}$$

Usually, appropriate values for η and the ρ_u will not be known *a priori*. Instead, we can give prior distributions to these hyperparameters. These priors can be fairly vague, but improper priors can cause problems.

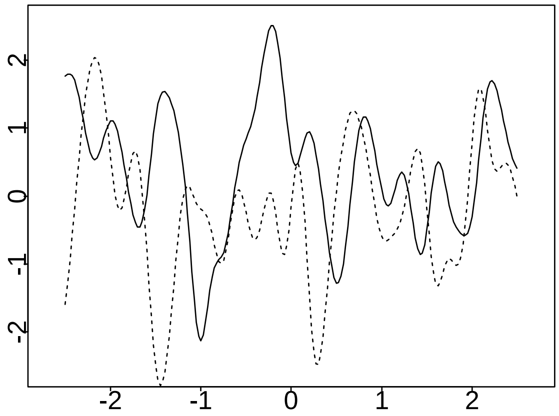
For the covariance function above, the ρ_u determine the relevance of each input to the prediction problem.

Some Functions Drawn From Univariate Gaussian Processes

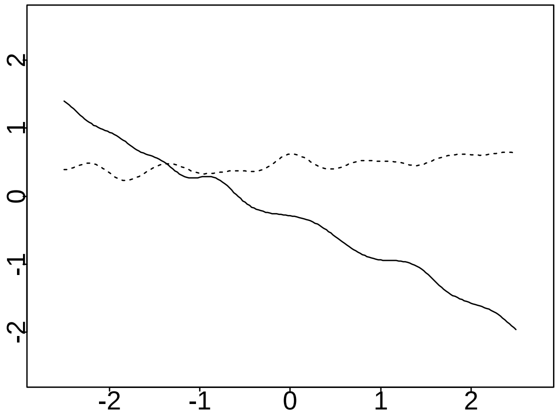
Each plot shows two functions randomly drawn from the Gaussian process with the given covariance function.



$\exp(-(x^{(i)} - x^{(j)})^2)$

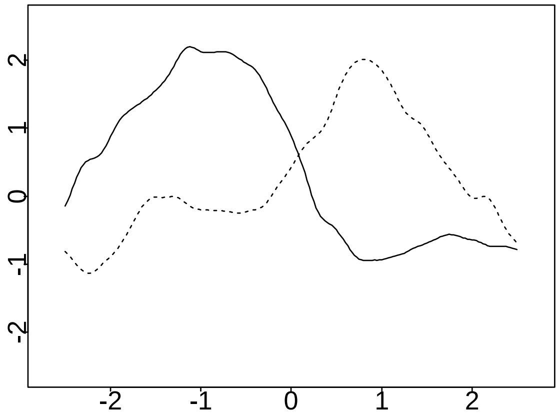


$\exp(-5^2(x^{(i)} - x^{(j)})^2)$



$1 + x^{(i)}x^{(j)}$

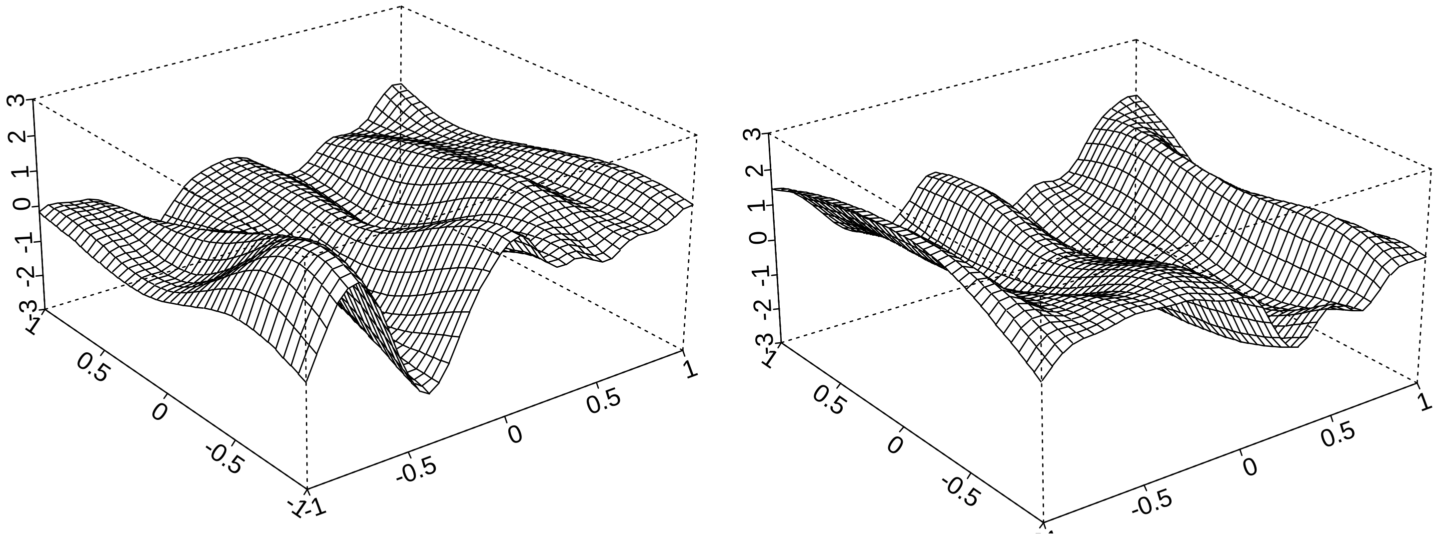
$+ 0.1^2 \exp(-3^2(x^{(i)} - x^{(j)})^2)$



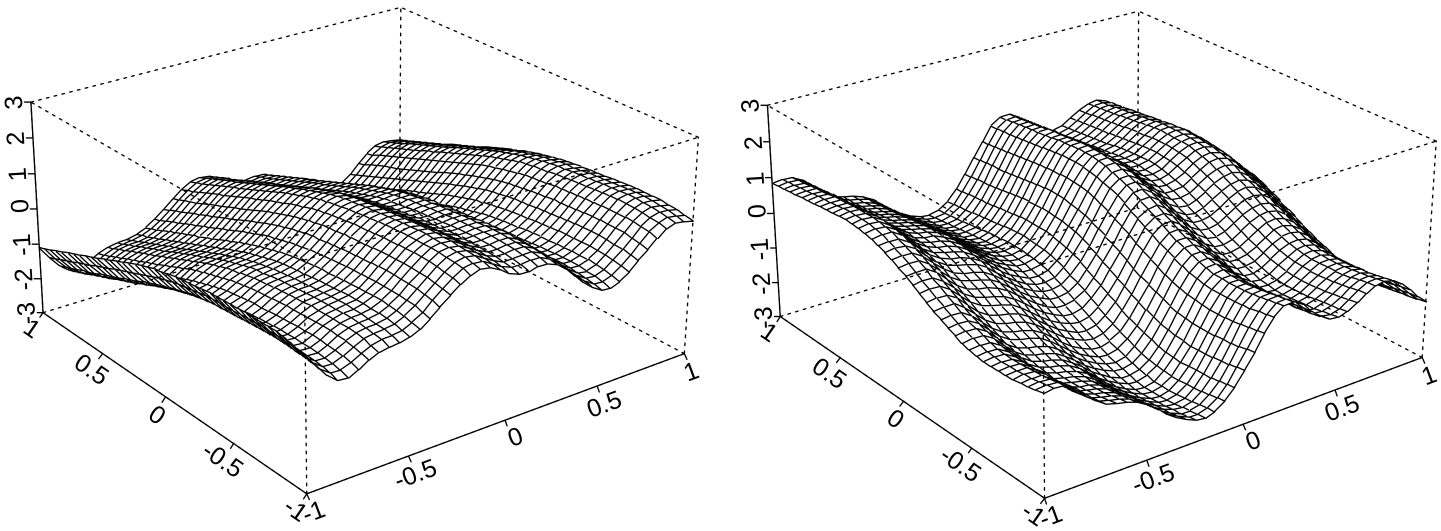
$\exp(-(x^{(i)} - x^{(j)})^2)$

$+ 0.1^2 \exp(-5^2(x^{(i)} - x^{(j)})^2)$

Some Functions Drawn From Bivariate Gaussian Processes



$$\text{Cov}\left[t^{(i)}, t^{(j)}\right] \;=\; \exp\left(-1.14^2(x_1^{(i)}-x_1^{(j)})^2 - 4.24^2(x_2^{(i)}-x_2^{(j)})^2\right)$$



$$\text{Cov}\left[t^{(i)}, t^{(j)}\right] \;=\; \exp\left(-1.14^2(x_1^{(i)}-x_1^{(j)})^2\right) + \exp\left(-4.24^2(x_2^{(i)}-x_2^{(j)})^2\right)$$

Making Predictions with a Gaussian Process Regression Model

We wish to predict the target, $t^{(n+1)}$, for a new case with inputs $x^{(n+1)}$, based on the inputs and targets for n observed cases.

Since everything is multivariate Gaussian, the conditional distribution for $t^{(n+1)}$ is Gaussian, with

$$\begin{aligned}
\mathsf{E}\left[t^{(n+1)}\,\middle|\,t^{(1)},\ldots,t^{(n)}\right] &= \mathbf{k}^T\,\mathbf{C}^{-1}\,\mathbf{t} \\
\mathsf{Var}\left[t^{(n+1)}\,\middle|\,t^{(1)},\ldots,t^{(n)}\right] &= V \;-\; \mathbf{k}^T\,\mathbf{C}^{-1}\,\mathbf{k}
\end{aligned}$$

Here, $\mathbf{t}$ is the vector of targets in the observed cases, $t^{(1)},\ldots,t^{(n)}$, and $\mathbf{C}$ is the covariance matrix for $\mathbf{t}$. The vector $\mathbf{k}$ consists of covariances between $t^{(n+1)}$ and $t^{(1)},\ldots,t^{(n)}$. Finally, V is the prior variance of $t^{(n+1)}$.

This takes time proportional to n^3.

Predictions for Classification Models

For a binary classification problem, the probability that a new case will be of class 1 is:

$$E\Big[\ [1+\text{exp}(-y^{(n+1)})]^{-1}\ \Big|\ t^{(1)},\ \ldots,\ t^{(n)}\ \Big]$$

We can estimate this probability by averaging over a sample of values for $y^{(n+1)}$, which has a Gaussian distribution given $y^{(1)},\ldots,y^{(n)}$.

We also need to average over the distribution of $y^{(1)},\ldots,y^{(n)}$ given $t^{(1)},\ldots,t^{(n)}$ (along with the $x^{(i)}$ and the hyperparameters). This can be done by Markov chain sampling.

Gibbs Sampling for Latent Values

We can sample for latent values by Gibbs sampling. The conditional distribution for one $y^{(i)}$ is proportional to the product of

- Its conditional density given $y^{(j)}$ for $j \neq i$. This is Gaussian, with mean and variance given in terms of $\mathbf{C}^{-1}$.

- Its likelihood based on the observed target, $t^{(i)}$. This is log-concave.

The final conditional distribution is also log-concave, so we can use Adaptive Rejection Sampling (Gilks and Wild, 1992).

Once $\mathbf{C}^{-1}$ is computed, a Gibbs sampling scan takes time proportional to n^2. We therefore might as well do at least around n scans between each update of the hyperparameters.

Metropolis-Hastings Updates

Gibbs sampling is slow when the jitter is small.

We can instead do a symmetric Metropolis update based on adding a scaled-down sample from the prior. We propose changes to the vector $\mathbf{y}$ of latent variables as follows:

$$\mathbf{y}' \;=\; \mathbf{y} \;+\; \epsilon L \mathbf{z}$$

where $\mathbf{L}$ is the Cholesky decomposition of $\mathbf{C}$, $\mathbf{z}$ is a vector of independent standard Gaussian variates, and ϵ is an adjustable stepsize. We accept or reject the proposal based on the product of prior and likelihood.

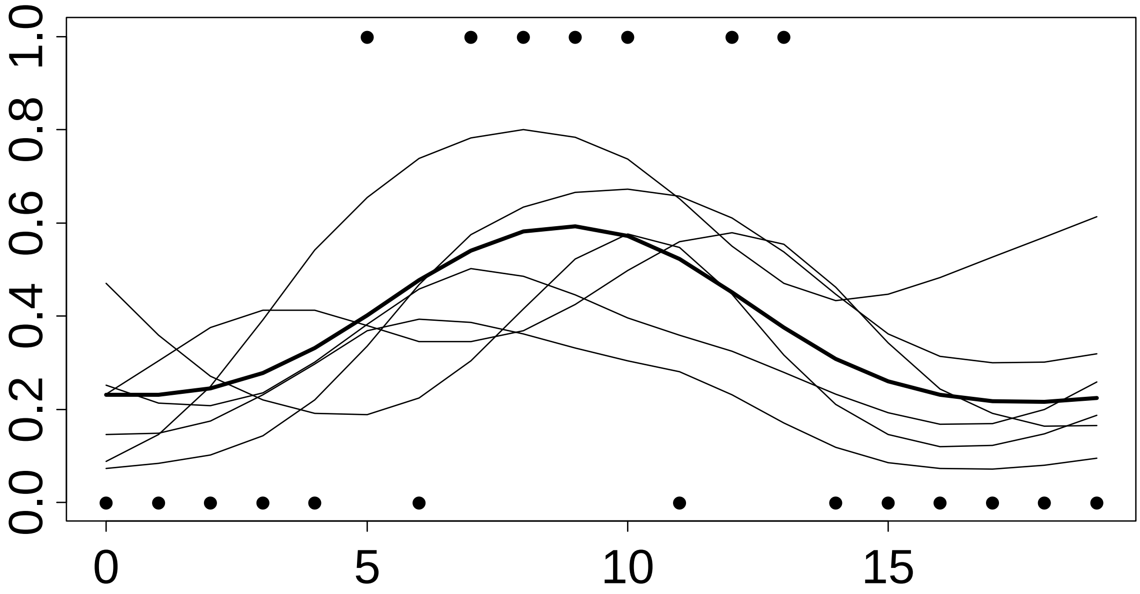
Better results are obtained with the proposal

$$\mathbf{y}' \;=\; (1-\epsilon^2)^{1/2}\,\mathbf{y} \;+\; \epsilon L \mathbf{z}$$

Since this proposal satisfies detailed balance w.r.t. to the prior, we accept or reject it based only on the likelihood.

Latent Values for a Simple Binary Classification Task

The dots below are 19 data points for a binary simple classification task with one input variable:



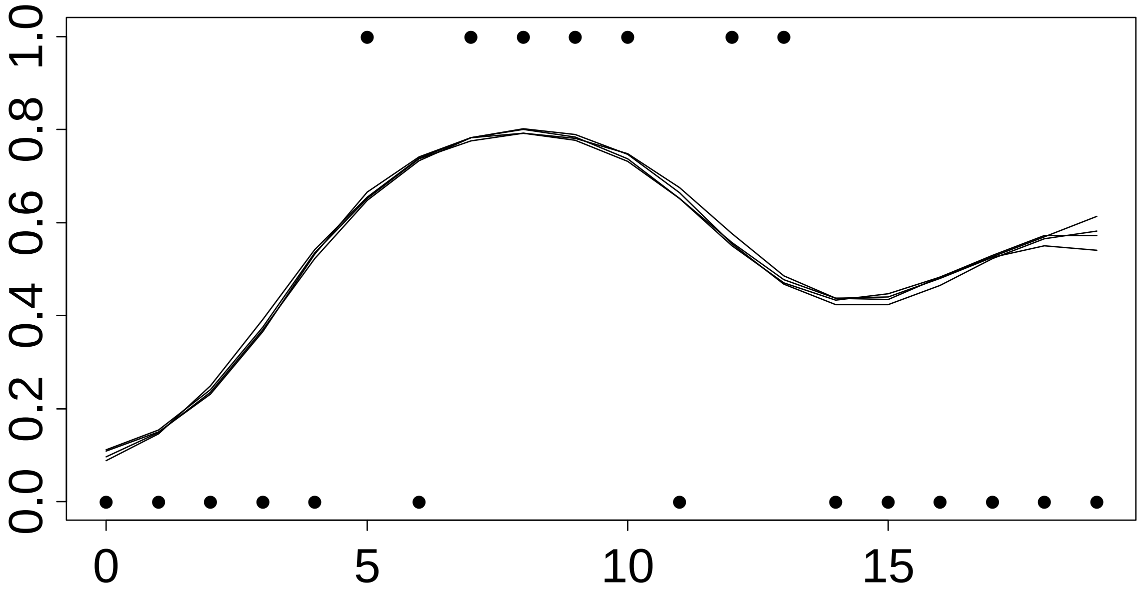
The light lines give class probabilities derived from six sets of latent values drawn from the posterior distribution, based on the following prior covariance function:

$$\text{Cov}[y^{(i)}, y^{(j)}] = 1 + \exp\left(-0.2^2(x^{(i)} - x^{(j)})^2\right) + 0.01^2 \delta_{ij}$$

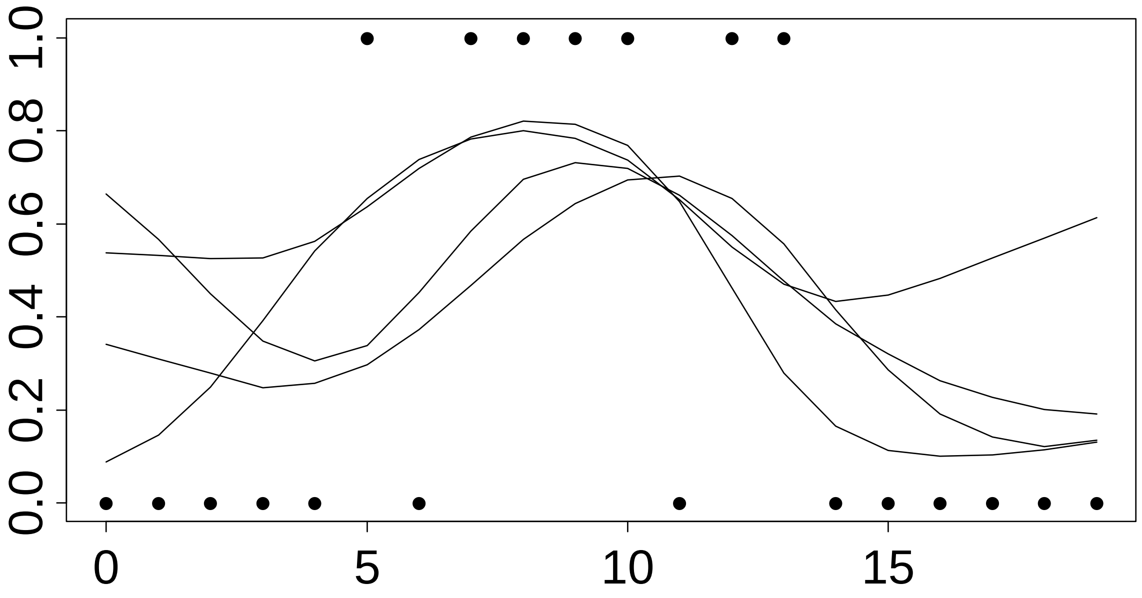
The dark line is the predictive distribution, averaging over the posterior.

Gibbs Sampling vs. Metropolis-Hastings

Four class probability functions from a Gibbs sampling chain, 10 iterations apart:



From a Metropolis-Hastings chain ($\epsilon = 0.7$):



Sampling for Hyperparameters

Hyperparameters can be updated using simple forms of the Metropolis algorithm, or by more sophisticated dynamical methods.

We need to compute the log likelihood for the hyperparameters:

$$L \;=\; -\,\frac{1}{2}\log\det\mathbf{C} \;-\; \frac{1}{2}\mathbf{t}^T\mathbf{C}^{-1}\mathbf{t}$$

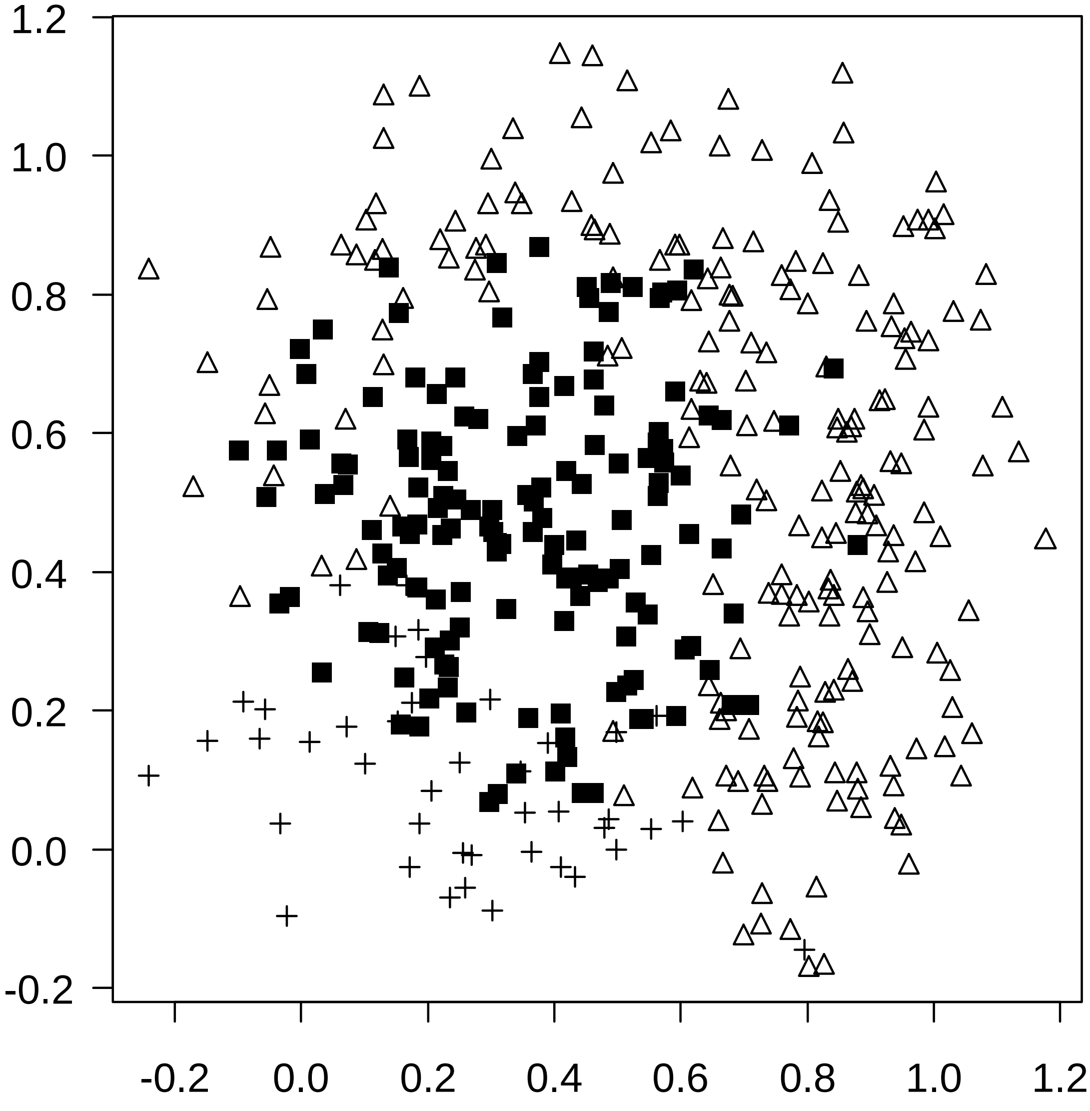
For classification models, we replace the targets, $\mathbf{t}$, by latent values, $\mathbf{y}$. We can computed L in time proportional to n^3 using the Cholesky decomposition of $\mathbf{C}$.

Dynamical methods also require the derivative of L with respect to the hyperparameters:

$$\frac{\partial L}{\partial \theta} \;=\; -\frac{1}{2}\,\mathrm{tr}\left(\mathbf{C}^{-1}\,\frac{\partial \mathbf{C}}{\partial \theta}\right) \;+\; \frac{1}{2}\,\mathbf{t}^T\,\mathbf{C}^{-1}\,\frac{\partial \mathbf{C}}{\partial \theta}\,\mathbf{C}^{-1}\,\mathbf{t}$$

Once $\mathbf{C}^{-1}$ has been computed, this takes time proportional to n^2 for each hyperparameter.

A Three-Way Classification Example



The 400 synthetically-generated cases used to fit the model are plotted above, according to the values of x_1 and x_2. Inputs x_3 and x_4 were also present, but were irrelevant.

The Gaussian Process Model Used

Covariance function for the latent variables:

$$\text{Cov}\,[y_k^{(i)}, y_k^{(j)}] \;=\; 10^2 + 10^2\,\delta_{ij} \\ \quad + \;\eta^2 \exp\Big(-\sum_{u=1}^{4} \rho_u\,(x_u^{(i)} - x_u^{(j)})^2\Big)$$

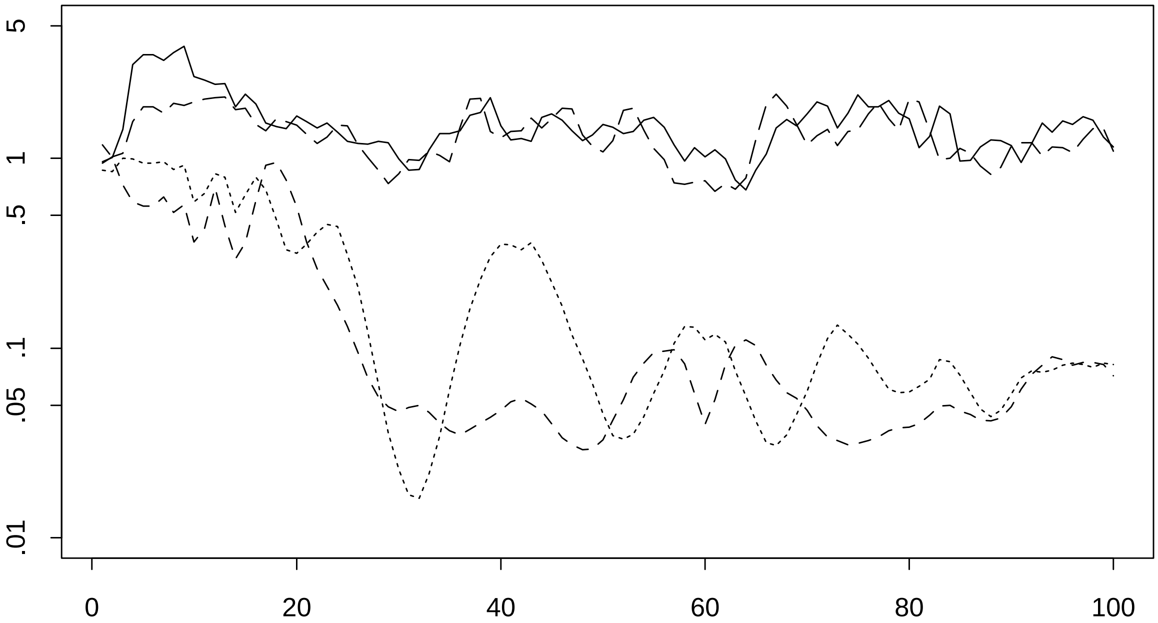
Priors for η and the ρ_u are vague, but proper.

The ρ_u hyperparameters control the relevance of the four inputs. We hope ρ_3 and ρ_4 will have a posterior distribution near zero, so that these irrelevant inputs will be ignored.

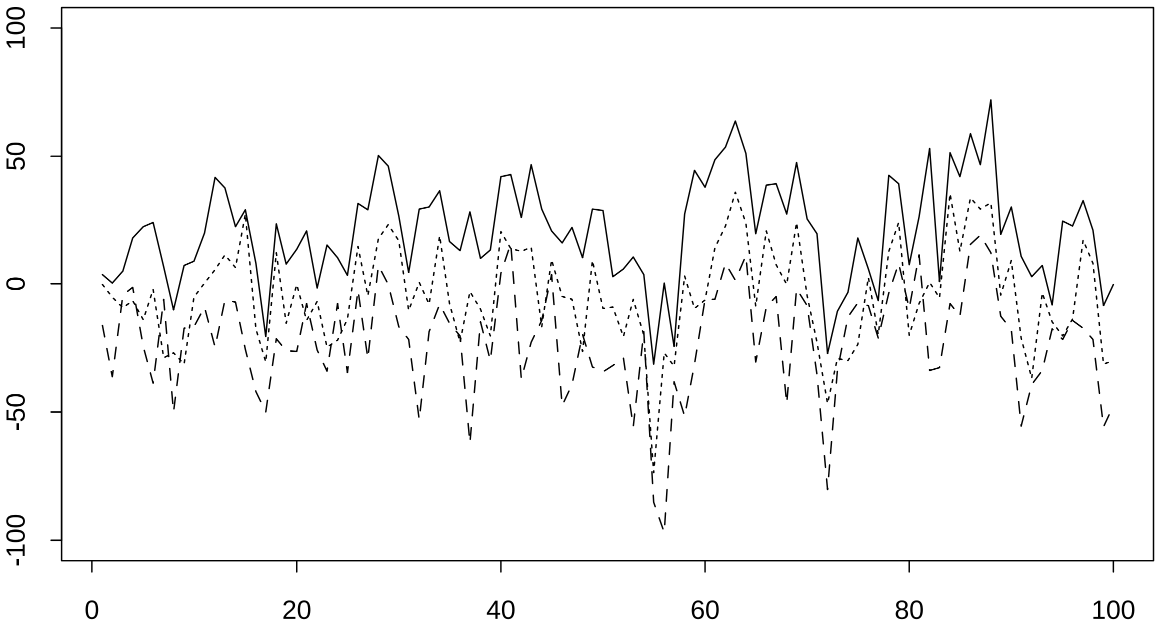
The $10^2\,\delta_{ij}$ term introduces quite a large amount of "jitter", producing the effect of a probit model.

Convergence of the Markov Chain

Several Hybrid Monte Carlo and Gibbs Sampling updates were done each iteration.



Progress of the four relevance hyperparameters



The latent values associated with one case

Results for this Example

- The time required for 100 iterations was about 220 minutes on a 200 MHz SGI machine. The Markov chain seems to have mixed fairly well during that time.

- The hyperparameters controlling the relevance of the four inputs adapted well to the true situation.

- The classification error rate on 600 new test cases was 13%. This seems pretty good. It is close to the rate for a Bayesian neural network model. The S-Plus “tree” procedure gives an error rate of 18%.

Previous Uses of Gaussian Processes

Gaussian processes arise in many contexts, and are well known.

So it is odd that they are not already widely used as Bayesian regression models! O'Hagan (1978) is one of the few to do so previously.

Other people have done related things:

- "Kriging" in spatial statistics is essentially Gaussian process regression — just not thought of that way.
- Smoothing splines are a special case of Gaussian process regression, but the Bayesian aspect is usually suppressed.
- Some work has been done on using Gaussian processes to model noise-free data from "computer experiments".
- Some work on "Bayesian numerical analysis" is based on Gaussian processes.

Work on Gaussian Process Models Derived from Neural Networks

I have shown how Bayesian neural network models reach limiting stochastic processes as the number of “hidden units” goes to infinity.

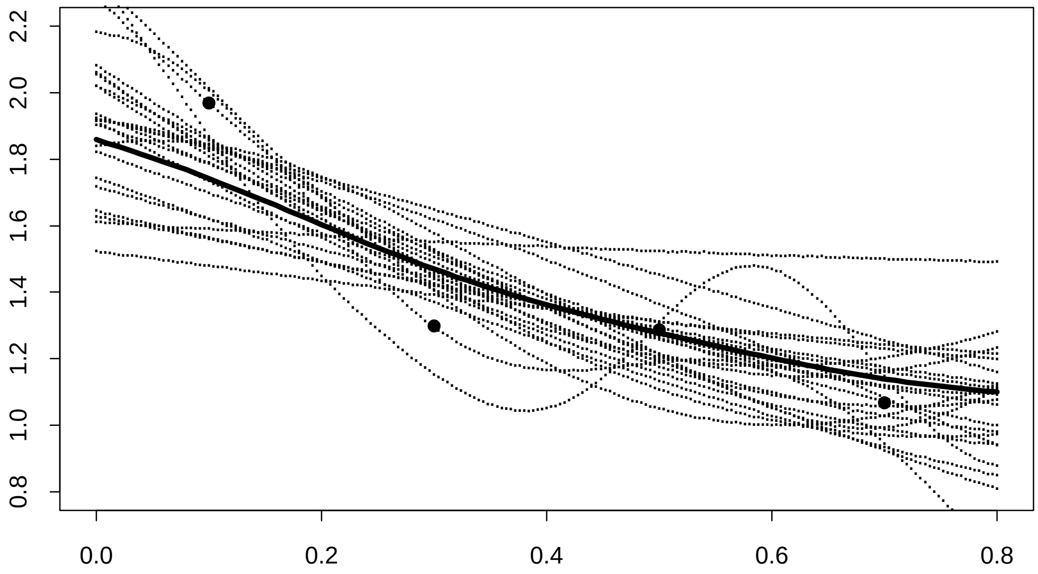
When the priors used have finite variance, the limit is a Gaussian process. Williams (1997) explicitly finds the covariance functions for some neural network models.

This has inspired work on Gaussian process regression and classification by Williams and Rasmussen (1996), Barber and Williams (1997), Gibbs and MacKay (1997), and myself (Neal 1997).

Why Aren't Gaussian Process Models Used More?

Perhaps it's due to confusion between the properties of the unknown function and the best predictor for that function.

Here are four data points and samples from the posterior for a Gaussian process model:



The posterior mean (solid) looks much simpler than many of the samples from the posterior.

Intuitions that the best predictor should be simple may lead to inappropriately simple models for the actual function.