CSC 2541: Bayesian Methods for Machine Learning

Radford M. Neal, University of Toronto, 2011

Lecture 8

Gaussian Process Models for Classification

A Gaussian process regression model for data $(x_1, y_1), \ldots, (x_n, y_n)$ can be expressed as

$$\begin{aligned} \sigma &\sim \ldots \\ \theta &\sim \ldots \\ f &\sim GP(\theta) \\ y_i \,|\, x_i,\, f &\sim N(f(x_i), \sigma^2) \end{aligned}$$

where θ represent all the parameters of the Gaussian process's covariance function. The Gaussian error combines with the Gaussian process for f to allow computations (for given σ and θ) to be done by matrix operations, integrating over f.

When the y_i are binary, we can use the following model:

$$\begin{aligned} \theta &\sim \ldots \\ f &\sim GP(\theta) \\ y_i \,|\, x_i,\, f &\sim \text{Bernoulli}\left(1 \,/\, (1 + \exp(-f(x_i)))\right) \end{aligned}$$

However, with this model, we cannot use simple matrix operations to evaluate $P(y|x, \theta)$, or to predict a new y^* by $P(y^*|x^*, y, x, \theta)$.

The Latent Gaussian Process

To do computations for Gaussian process classification, we need to explicitly represent the "latent variables" $z_i = f(x_i)$. For numerical reasons, we'll actually let z_i be $f(x_i)$ plus a small amount of "jitter", with variance j^2.

The model with latent variables can be expressed as

$$\begin{aligned} \theta &\sim \ldots \\ f &\sim GP(\theta) \\ z_i \,|\, x_i,\, f &\sim N(f(x_i), j^2) \\ y_i \,|\, z_i &\sim \text{Bernoulli}\,(1 \,/\, (1 + \exp(-z_i))) \end{aligned}$$

Using matrix operations, we can compute the joint density of the latent variables and observed responses (for given θ):

$$\begin{aligned} & P(z_1, \ldots, z_n, y_1, \ldots, y_n \,|\, x_1, \ldots, x_n, \theta) \\ &= \; P(z_1, \ldots, z_n \,|\, x_1, \ldots, x_n, \theta)\, P(y_1, \ldots, y_n \,|\, z_1, \ldots, z_n) \end{aligned}$$

The first factor above (the prior for latent variables) is Gaussian; the second (the likelihood for these latent variables) is a simple product of Bernoulli probabilities.

Handling the Latent Variables

Two methods are commonly used for handling the latent variables:

Approximate their posterior distribution by a Gaussian: The prior for $z_1, \ldots, z_n$ given θ is Gaussian. The likelihood, $P(y_1, \ldots, y_n \mid z_1, \ldots, z_n)$, is not, but as $n \to \infty$ it will approach a Gaussian form. Maybe a Gaussian posterior for $z_1, \ldots, z_n$ will be adequate for finite n.

Sample for the latent variables using MCMC: Conditional on θ, we do some Markov chain update of $z_1, \ldots, z_n$, that leaves the posterior distribution invariant. We also do updates for θ, and base predictions on averaging over the samples we obtain for z and θ.

Gibbs sampling for $z_1, \ldots, z_n$ is possible (using "Adaptive Rejection Sampling"). However, often there will be z_i and $z_{i'}$ that are highly dependent, so one Gibbs sampling scan will make only very small changes. We'd like a better way...

A Metropolis-Hastings Update for the Latent Variables

We can propose a change to all the latent variables at once that respects their correlation structure, as follows:

$$z^* \quad = \quad (1 - \epsilon^2)^{1/2} z \; + \; \epsilon L n$$

where n is a vector of independent $N(0, 1)$ variates and L is the Cholesky decomposition of the prior covariance matrix, C, of the latent variables (ie, $C = LL^T$).

We use the usual Metropolis-Hastings acceptance probability for this,

$$\min \left[1, \; \frac{\ell(z^*)\pi(z^*)S(z^*, z)}{\ell(z)\pi(z)S(z, z^*)} \right]$$

where π is the prior, ℓ is the likelihood, and S is the proposal density.

One can show that for the proposal above, $\pi(z)S(z, z^*) = \pi(z^*)S(z^*, z)$, so the acceptance probability simplifies to $\min(1, \ell(z^*)/\ell(z))$.

We need to choose a suitable value for ϵ in $(-1, 1)$. This can be avoided using "Elliptical Slice Sampling" (Murray, Adams, and MacKay, 2010).

Hierarchical Bayesian Models

Complex Bayesian models are often specified using a hierarchy, of observable variables, latent variables, parameters, and hyperparameters at higher levels.

Latent variables / random effects: To model time to recovery from a certain kind of surgery, we introduce a mean time for each surgeon, a mean of these mean times for surgeons in each hospital, a mean of these means of means for each city, etc.

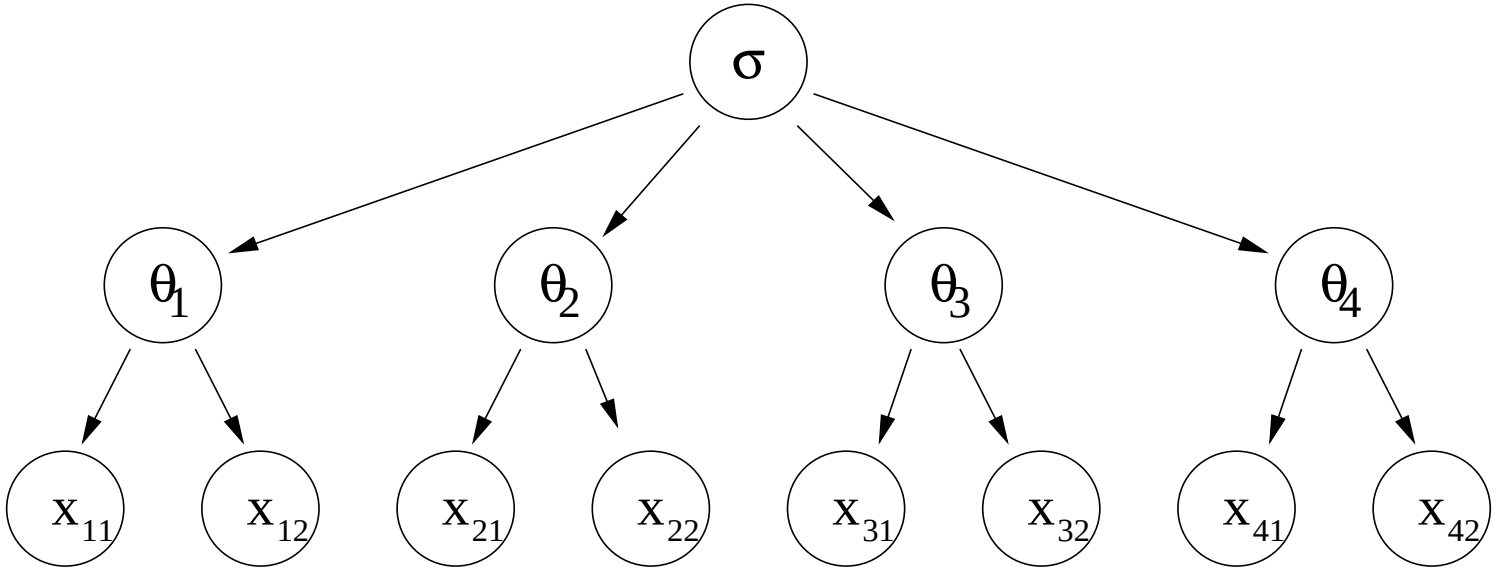
Structured prior distributions: In a model for classifying genes by function, we introduce hyperparameters representing the importance of different sources of data (eg, gene expression data versus gene sequence data), which control the priors of parameters relating to variables from each source.

The distinction between latent variables and parameters is vague, and is not crucial from a Bayesian perspective. But for frequentists, it can be a big issue whether or not something should be regarded as a "random effect", since these, but not parameters (eg, "fixed effects"), are modeled probabilistically.

Representing Dependencies by a Directed Graphical Model

How the observable variables depend on higher levels in the hierarchy can be expressed using a directed graphical model.

For example:



This says that the joint density of σ, $\theta_1, \ldots, x_{11}, \ldots$ can be written as

$$P(\sigma) \prod_i P(\theta_i|\sigma) \prod_{i,j} P(x_{ij}|\theta_i)$$

The absense of an arrow indicates that an earlier variable need not be conditioned on in this expression — eg, that $P(x_{11}|\theta_1) = P(x_{11}|\theta_1, \sigma)$.

The Same Model as Formulas

The following notation expresses the same dependence relationships, along with the actual form of the distributions:

$$\begin{aligned} \sigma &\sim \exp(1) \\ \theta_i|\sigma &\sim N(0, \sigma^2) \\ x_{ij}|\theta_i &\sim N(\theta_i, 1) \end{aligned}$$

In this notation also, there is some ordering of variables, and their distributions are show conditional on earlier variables, omitting earlier variables that are conditionally independent given the ones that are shown.

Integrating Away Some Parameters / Hyperparameters

In principle, we can always integrate away any variable that is not observable, giving a model with fewer variables, but perhaps more complex distributions.

For the model

$$\begin{aligned} \sigma &\sim \exp(1) \\ \theta_i|\sigma &\sim N(0, \sigma^2) \\ x_{ij}|\theta_i &\sim N(\theta_i, 1) \end{aligned}$$

we can integrate out the θ_i parameters, giving

$$\begin{aligned} \sigma &\sim \exp(1) \\ x_i|\sigma &\sim N(0, \Sigma) \end{aligned}$$

where $x_i = [x_{i1},\ x_{i2}]$ and

$$\Sigma = \begin{bmatrix} 1+\sigma^2 & \sigma^2 \\ \sigma^2 & 1+\sigma^2 \end{bmatrix}$$

But trying to integrate away σ too doesn't produce a tractable result.

Why Introduce Hyperparameters?

We see that hyperparameters are in theory unnecessary — we can always define the prior distribution for the lower-level parameters directly.

But hyperparameters are a convenient way of introducing *dependence* among parameters, with these parameters being *independent* given the value of the hyperparameters.

From this viewpoint,

- There's not much point in introducing a hyperparameter that controls just one parameter.
- Whenever you have more than one similar parameter, you should probably introduce a hyperparameter to allow for them being dependent.

For complex models, hyperparameters can also allow learning of high-level properties of the data — eg, the degree of smoothness of a function, and whether or not it has an additive form.

Learning Covariance Functions for Gaussian Process Models

Gausian processes allow for a rich variety of priors over functions, as determined by the covariance function. Since we usually don't know which covariance function is appropriate, we define a class of covariance functions with unknown parameters, give prior distributions to these parameters, and learn them from the data. These parameters include the noise variance as well, for regression models.

Parameters of the covariance function may also be called "hyperparameters", since they would control lower-level parameters if we expressed the model using explicit basis functions. There may be higher level hyperparameters controlling these parameters/hyperparameters.

We can visualize the properties of different classes of covariance functions by looking at functions randomly drawn from a Gausian process with such a covariance function.

Randomly Generating a Function from a Gaussian Process

Plotting a 1D or 2D function randomly drawn from a Gaussian process is easy.

We first produce a grid of x values at which we need function values in order to produce our plot — say $x_1, \ldots, x_n$.

We then find the covariance matrix for the associated function values, $y_1, \ldots, y_n$, call it C. To avoid numerical difficulties, we may have to add a small amount to the diagonal of C, corresponding to the y_i being the function value at x_i plus a small amount of Gaussian noise.

We then find the Cholesky decomposition of C — the lower-triangular matrix, L, such that $C = LL^T$.

Finally, we generate a random vector, n, of independent Gaussian variates with mean 0 and variance 1, and compute $y = [y_1, \ldots, y_n]^T$ as $y = Ln$.

This y will be Gaussian, with mean 0, and covariance of

$$E[yy^T] = E[(Ln)(Ln)^T] = E[Lnn^TL^T] = LE[nn^T]L^T = LL^T = C$$

Learning the Scale of Variation

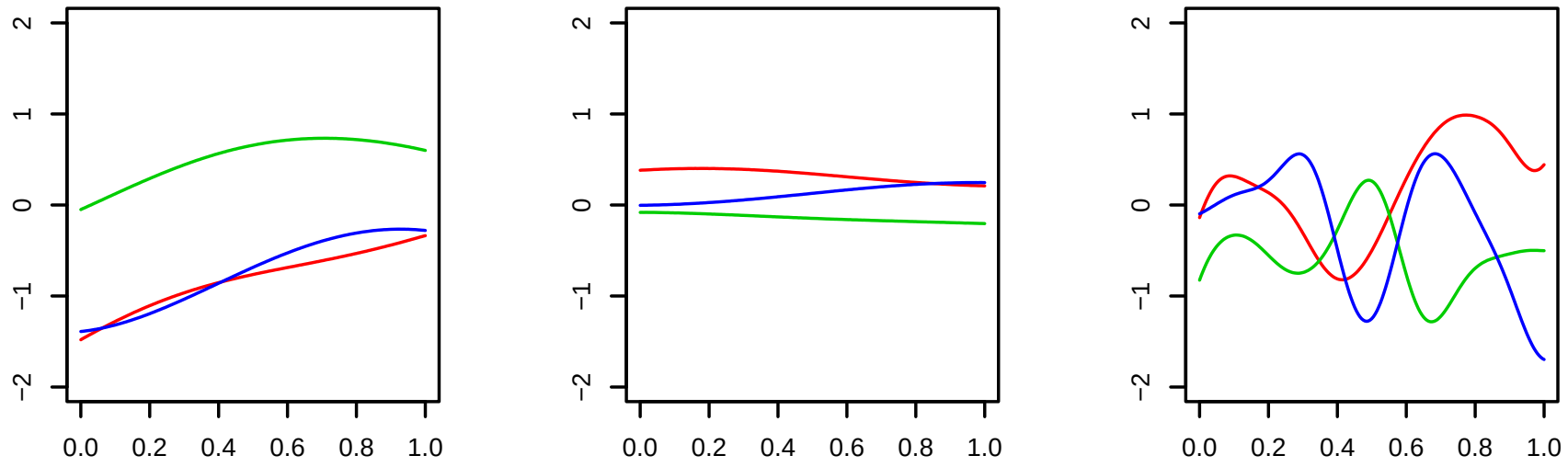
Consider a covariance function for a single input, x, of the form

$$K(x, x') \quad = \quad \gamma^2 \exp(-\rho^2 (x - x')^2)$$

The γ parameter controls the vertical scale of functions drawn according to the covariance function, while the ρ parameter controls the horizontal scale.

Different γ and ρ, three random functions for each:

$\gamma = 1, \ \rho = 1$ $\gamma = 0.2, \ \rho = 1$ $\gamma = 1, \ \rho = 6$



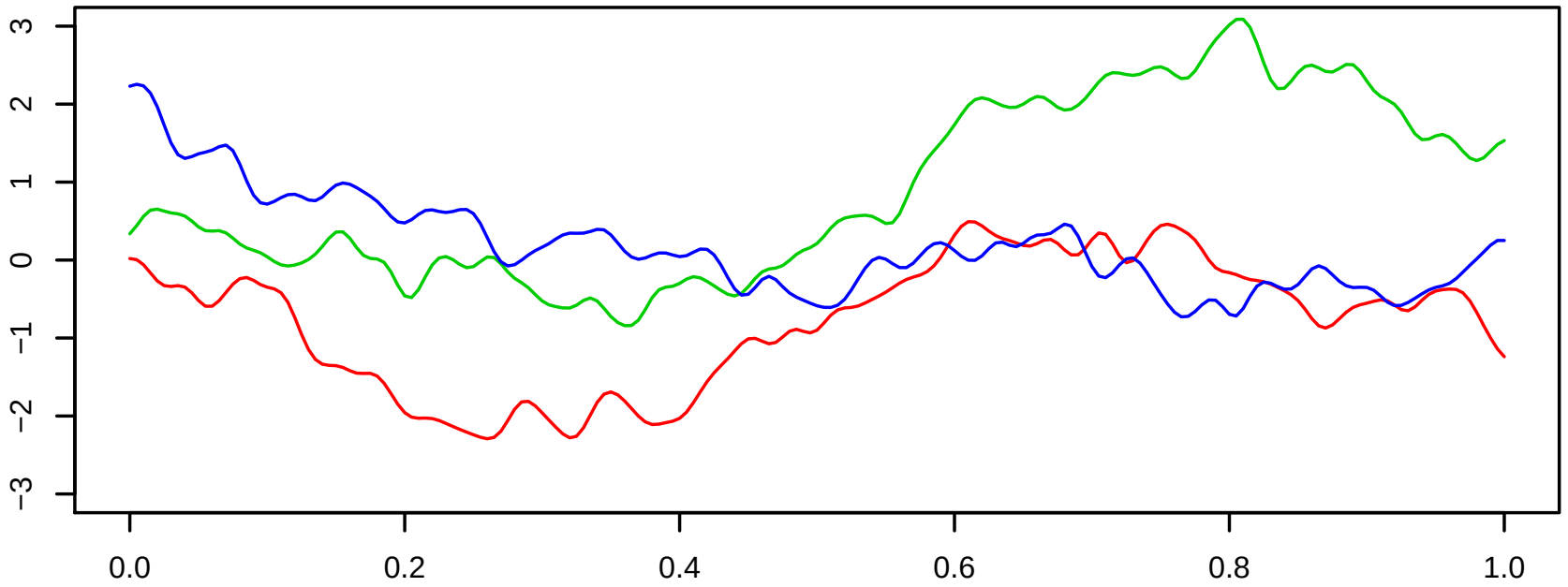
We usually don't know the appropriate scales, so we let γ and ρ be unkown, with some priors. Overly broad priors should be avoided, but priors spanning several factors of ten are OK, if we don't have more specific information.

Multiple Scales of Variation

We can obtain multiple scales of variation by adding terms with different scale parameters. For example:

$$K(x, x') \;=\; \gamma_A^2 \exp(-\rho_A^2 (x - x')^2) \;+\; \gamma_B^2 \exp(-\rho_B^2 (x - x')^2)$$

Functions drawn from such a process with $\gamma_A = 1,\ \rho_A = 4,\ \gamma_B = 0.2,\ \rho_B = 50$:



By letting γ_A, γ_B, ρ_A, and ρ_B be unknown, we can learn whether such multiple scales are appropriate, and if so what the parameters should be. Multiple scales will be elminated if either γ_A or γ_B is close to zero (or, less obviously, if either ρ_A or ρ_B is close to zero).

Learning the Relevance of Inputs

Similar classes of covariance functions can be used with $p > 1$ inputs. We can choose between using the same scale for all inputs, for example:

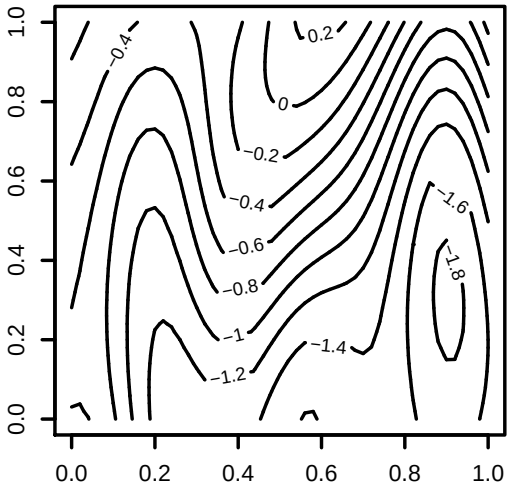
$$K(x, x') = \gamma^2 \exp\Big(-\rho^2 \sum_{j=1}^{p} (x - x')^2 \Big)$$

or using a different scale for each input, for example:

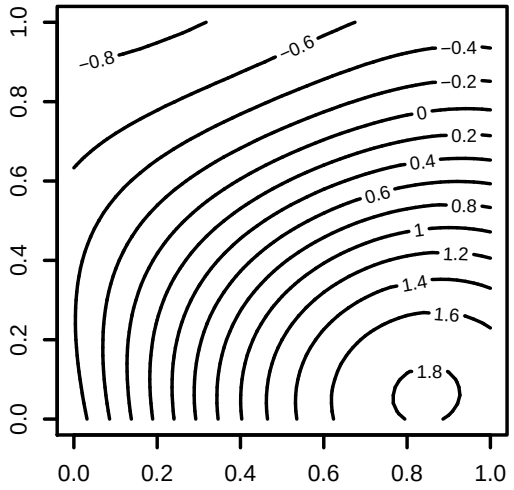
$$K(x, x') = \gamma^2 \exp\Big(-\sum_{j=1}^{p} \rho_j^2 (x - x')^2 \Big)$$

Learning ρ_j from data is sometimes called "Automatic Relevance Determination". Examples with two inputs:

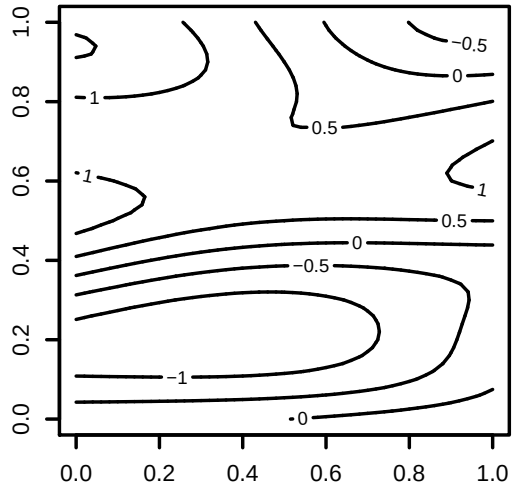
$\rho_1 = 4,\ \rho_2 = 1$



$\rho_1 = 1,\ \rho_2 = 1$



$\rho_1 = 1,\ \rho_2 = 4$



Hierarchical Relevance

Automatic Relevance Determination with many inputs requires many relevance hyperparameters, $\rho_1, \ldots, \rho_p$. We probably don't know what prior to give them, so we may introduce a higher-level hyperparameter, ρ_0. For example:

$$\begin{aligned} \log(\rho_0) &\sim N(0, 5^2) \\ \log(\rho_j) \,|\, \log(\rho_0) &\sim N(\log(\rho_0), 3^2), \quad \text{for } j = 1, \ldots, p \end{aligned}$$

This assumes that we have no prior reason to think one input is more likely to be relevant than another. If we have such knowledge, we can rescale the inputs in a pre-processing step so that the above prior is still appropriate.

The inputs might come in natural groups — eg, measurements of a subject's social status, physical health, and education. We might then use one higher-level hyperparameter for each group. For example:

$$\log(\rho_A) \sim N(0, 5^2), \quad \log(\rho_B) \sim N(0, 5^2), \quad \log(\rho_C) \sim N(0, 5^2)$$

$$\begin{aligned} \log(\rho_j) \,|\, \log(\rho_A) &\sim N(\log(\rho_A), 3^2), \quad \text{for } j \in A \\ \log(\rho_j) \,|\, \log(\rho_B) &\sim N(\log(\rho_B), 3^2), \quad \text{for } j \in B \\ \log(\rho_j) \,|\, \log(\rho_C) &\sim N(\log(\rho_C), 3^2), \quad \text{for } j \in C \end{aligned}$$

We might introduce a still-higher-level hyperparameter to control ρ_A, ρ_B, and ρ_C.

Covariance Functions for Additive and Interactive Models

With $p \geq 2$ inputs, we can define a covariance function that produces an additive function — a sum of a function of just x_1, plus a function of just x_2, etc.

We just add together covariance functions for each input. For example:

$$K(x, x') = \sum_{j=1}^{p} \gamma_j^2 \exp(-\rho_j^2 (x_j - x'_j)^2)$$

Compare with an interactive covariance function:

$$K(x, x') = \gamma^2 \exp(-\sum_{j=1}^{p} \rho_j^2 (x_j - x'_j)^2)$$

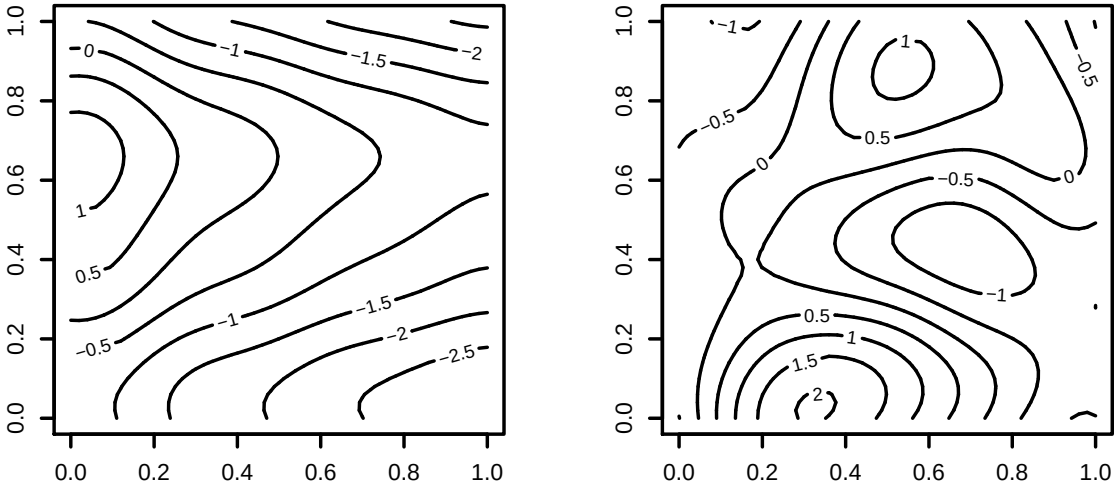
Examples with two inputs:

Left: additive

Right: interactive

$\gamma = \gamma_1 = \gamma_2 = 1$

$\rho_1 = \rho_2 = 3$



With a covariance function that's a sum of additive and interactive covariances, the data can determine which to use (or whether to use a combination).

Hierarchical Priors for Mixture Models

Consider this Gaussian mixture model for vectors of p real variables, with K components (perhaps with $K \to \infty$):

$$\begin{aligned}
\rho_1, \ldots, \rho_K &\sim \text{Dirichlet}\,(\alpha/K, \ldots, \alpha/K) \\
c_i \mid \rho_1, \ldots, \rho_K &\sim \text{Discrete}\,(\rho_1, \ldots, \rho_K) \\
\mu_c &\sim N(\mu_0, \text{diag}(\sigma_0^2)) \\
\log(\sigma_c) &\sim N(\nu, \text{diag}(\tau^2)) \\
y_i \mid c_i,\, \mu,\, \sigma &\sim N(\mu_{c_i}, \text{diag}(\sigma_{c_i}^2))
\end{aligned}$$

where μ_c and σ_c are vectors of means and standard deviations for the variables in mixture component c.

If we make μ_0, σ_0, ν, and τ be hyperparameters (with suitable priors), the model can discover which variables should be modeled differently for each component, and which should have (almost) the same distribution for all components (by letting σ_0 and τ for that variable be close to zero).

We probably also want α to be determined by the data.