CSC 2541: Bayesian Methods for Machine Learning

Radford M. Neal, University of Toronto, 2011

Lecture 5

Problem: Regression

We have observed pairs, $(x_1, y_1), \ldots, (x_n, y_n)$, where the x_i are given (their distribution not modeled), and the y_i are related to the x_i by some function, plus noise that is independent for each i.

The inputs/covariates/predictors x_i are numerical, and usually multidimensional (p predictors). The targets/responses y_i are typically real (though generalization to vector y_i is possible).

Possible approaches:

- Use a simple parametric model — eg, y_i is a linear function of x_i plus noise.
- Use a non-model-based method — eg, nearest-neighbor or more sophisticated local smoothing methods.
- Use a flexible model for the conditional distribution of y_i given x_i — eg, Gaussian processes, neural networks.

Possible difficulties:

- Noise that isn't Gaussian (or other simple form), or that has variance (or other aspects of its distribution) that depends on x.
- Possibility that $p \geq n$, or even that $p \gg n$. Common in bioinformatics.

Non-linear Relationships from Linear Models

We can try to get models that are flexible but still tractable by making the parameterized relationship of $E(y)$ to x be *non-linear in* x, but *linear in the parameters*.

We can get this by doing ordinary linear regression of y on $\phi_1(x), \ldots, \phi_{M-1}(x)$, where the ϕ_j are *basis functions*, that we have selected to allow for a non-linear function of x. Usually we also include $\phi_0(x) = 1$, to allow any constant offset.

This gives the following model:

$$y \,|\, x, \beta \quad \sim \quad N(\phi(x)^T \beta, \sigma^2)$$

where $\phi(x)$ is the vector of basis function values for inputs x and β is the vector of all M regression coefficients, $[\beta_0, \beta_1, \ldots, \beta_M]'$.

Selecting the Basis Functions

There are various options for selecting the *basis functions*, $\phi_1(x), \ldots, \phi_{M-1}(x)$.

Possibilities for basis functions include polynomials, sigmoidal functions, and "radial basis functions" that look like Gaussian density functions.

Some approaches:

- Choose basis functions manually, as appropriate for a particular problem (eg, based on scientific knowledge of the relationship).
- Use a large number of basis functions that "cover" the relevant range of x, with characteristics that are determined by a small number of parameters (eg, the width of a local basis function).
- Use a smaller number of basis functions that are adapted to the data — neural networks with one hidden layer.
- Use an infinite number of basis functions, which for a Bayesian model with Gaussian priors can be handled with finite computations by viewing the y_i as coming from a "Gaussian process".

Maximum Likelihood Estimation

Suppose we assume that the noise in the regression model is Gaussian (normal) with mean zero and some variance σ^2.

With this assumption, we can write down the likelihood function for the parameters β and σ, which is the joint probability density of all the y_i as a function of β and σ:

$$\begin{aligned} L(\beta, \sigma) &= P(y_1, \ldots, y_n | x_1, \ldots, x_n, \beta, \sigma) \\ &= \prod_{i=1}^{n} N(y_i | \beta^T \phi(x_i), \sigma^2) \end{aligned}$$

where $N(y|\mu, \sigma^2)$ is the density for y under a normal distribution with mean μ and variance σ^2.

The maximum likelihood estimates for β and σ maximize the log likelihood, which, ignoring terms that don't depend on β or σ, is

$$\log L(\beta, \sigma) = -n \log(\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \beta^T \phi(x_i))^2$$

Least Squares Estimation

From this log likelihood function, we see that regardless of what σ might be, the maximum likelihood estimate of β is the value that minimizes the sum of squared prediction errors over training cases.

Let's put the values of all basis functions in all training cases into a matrix Φ, with $\Phi_{ij} = \phi_j(x_i)$, and the responses for all training cases into a vector y.

We can now write the sum of squared errors on training cases as

$$||y - \Phi\beta||^2 \quad = \quad (y - \Phi\beta)^T(y - \Phi\beta)$$

This is minimized for the value of β where its gradient is zero, which is where

$$-2\Phi^T(y - \Phi\beta) = 0$$

Solving this, the least squares estimate of β is

$$\hat{\beta} \quad = \quad (\Phi^T\Phi)^{-1}\Phi^T y$$

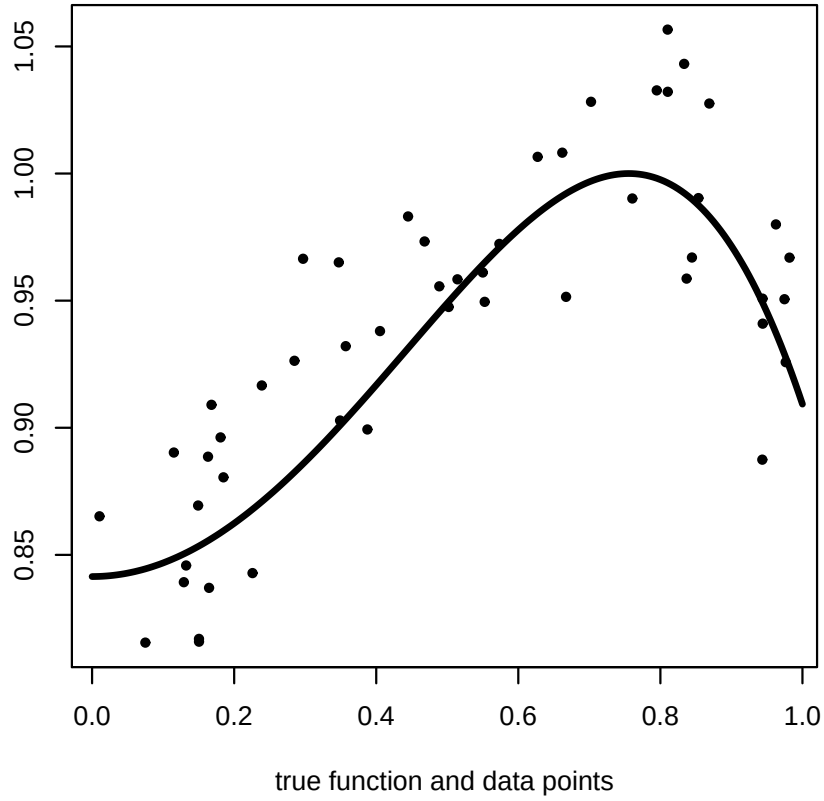
This assumes that $\Phi^T\Phi$ is non-singular, so that there is a unique solution. When M is greater than n, this will not be the case!

An Example Problem

For illustration, here is a synthetic data set of 50 observations, generated with x uniform from $(0, 1)$ and y set by the formula:

$$y \;=\; \sin(1 + x^2) \;+\; \text{noise}$$

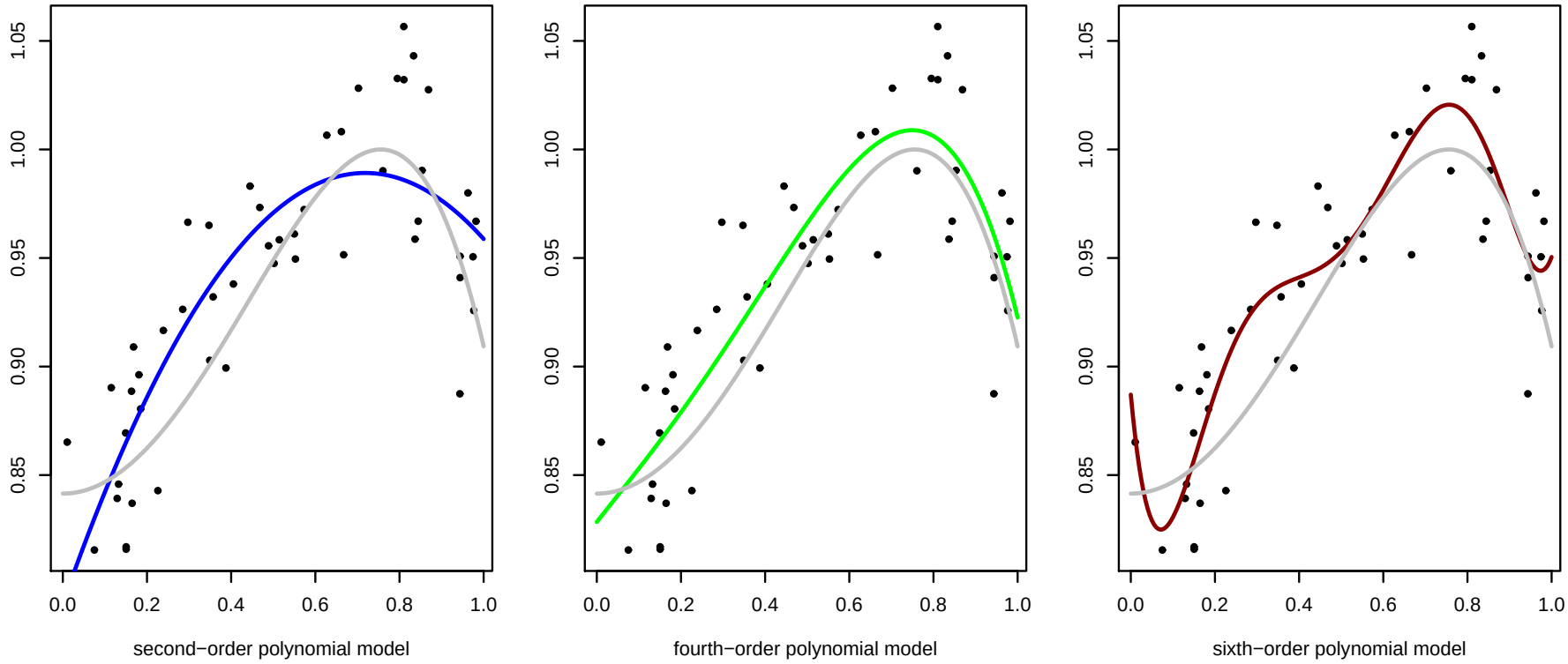
where the noise has $N(0, 0.03^2)$ distribution.



The noise-free true function, $\sin(1 + x^2)$, is shown by the line.

Least Squares with Polynomial Basis Functions

Here are least-squares fits with polynomial basis functions, $\phi_j(x) = x^j$.



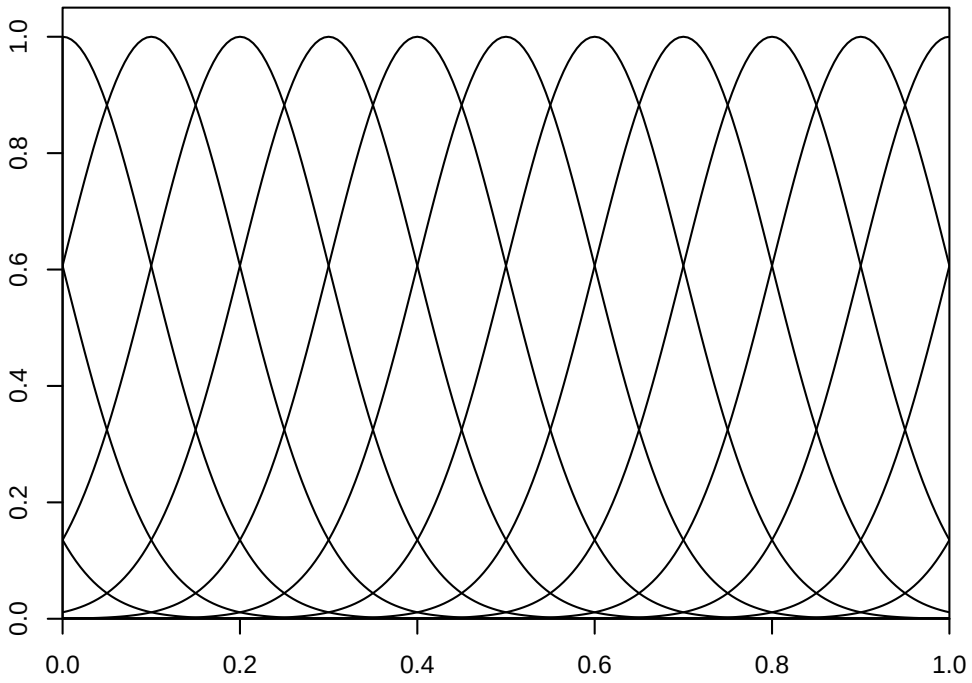
The gray line is the true noise-free function. We see that a second-order fit is too simple, but a sixth-order fit is too complex, producing "overfitting".

Gaussian Basis Functions

Polynomials are *global* basis functions, each affecting the prediction over the whole input space. Often, *local* basis functions are more appropriate. One possibility is to use functions proportional to Gaussian probability densities:

$$\phi_j(x) \quad = \quad \exp(-(x - \mu_j)^2/2s^2)$$

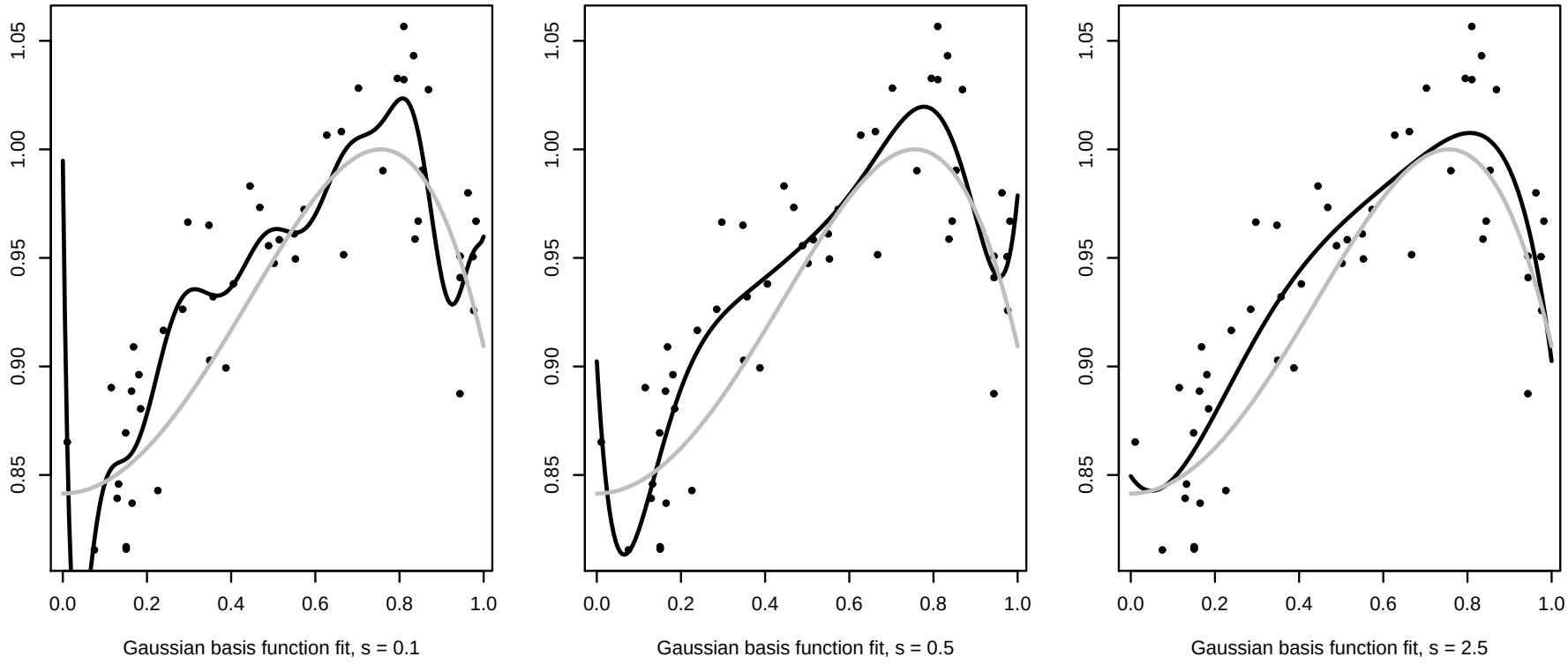
Here are these basis functions for $s = 0.1$, with the μ_j on a grid with spacing s:



Note that these basis functions have no real connection with Gaussian distributions.

Least Squares with Gaussian Basis Functions

Here are the results using Gaussian basis functions (plus $\phi_0(x) = 1$) on the example dataset, with varying width (and spacing) s:



The estimated values for the β_j are not what you might guess. For the middle model above, they are as follows:

6856.5 -3544.1 -2473.7 -2859.8 -2637.7 -2861.5 -2468.0 -3558.4

Maximum Penalized Likelihood Estimation

We can try to avoid the poor results of maximum likelihood when there are many parameters by adding a *penalty* to the log likelihood, that favours non-extreme values for the parameters. This procedure is also called *regularization*

For regression with Gaussian noise, we minimize the sum of squared errors on training cases plus this penalty.

Quadratic penalties are easiest to implement, as they combine with the squared error to still allow solution by matrix operations. For basis function models, we might use a penalty that encourages all β_j (except β_0) to be close to zero:

$$\lambda \sum_{j=1}^{M-1} \beta_j^2$$

Here, λ controls the strength of the penalty.

Solution for Penalized Least Squares

We found the least squares solution before by equating the gradient of the squared error to zero. Now we add the gradient of the penalty function as well, and hence solve

$$2\lambda\beta^* \; - \; 2\Phi^T(t - \Phi\beta) = 0$$

where β^* is equal to β except that β_0 is zero.

Solving this, the penalized least squares estimate of β is

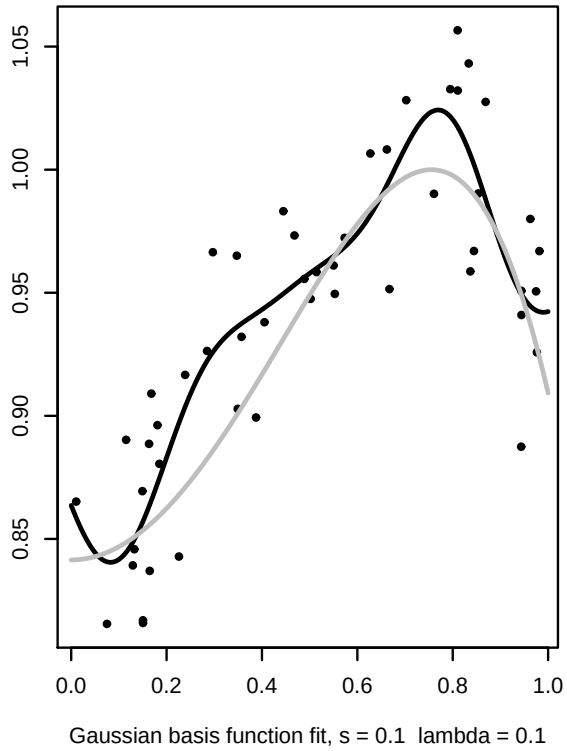
$$\hat{\beta} \;\; = \;\; (\lambda I^* + \Phi^T\Phi)^{-1}\Phi^T t$$

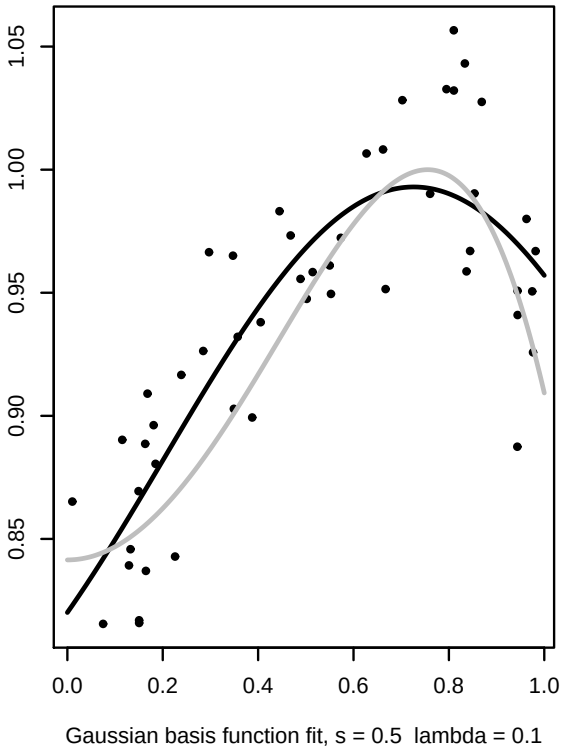
where I^* is like the identity matrix except that $I^*_{1,1} = 0$.

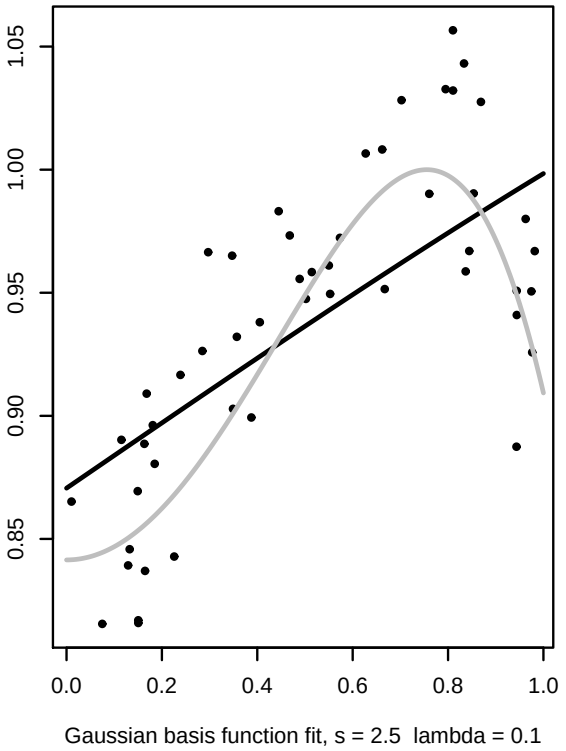
Note that this estimate will be uniquely defined regardless of how big M and n are, as long as λ is greater than zero.

Results with Regularized Gaussian Basis Functions

Here are the results with $\lambda = 0.1$:

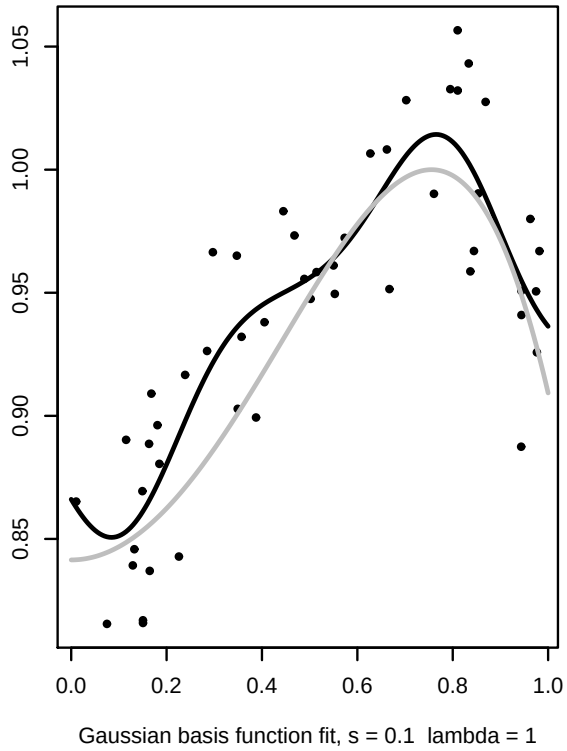


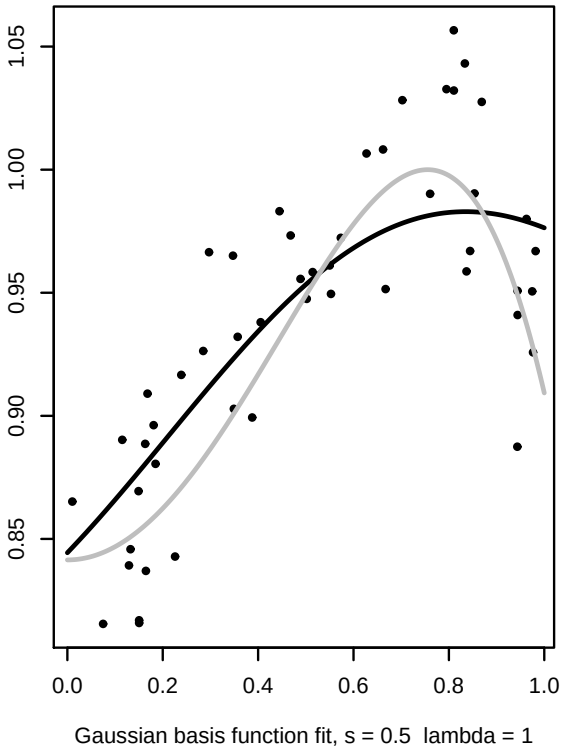


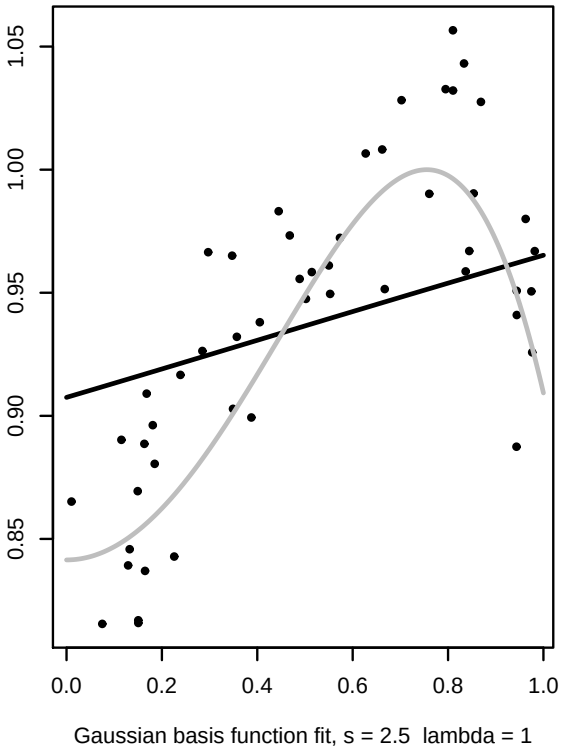


More Results with Regularized Gaussian Basis Functions

Here are the results with $\lambda = 1$:

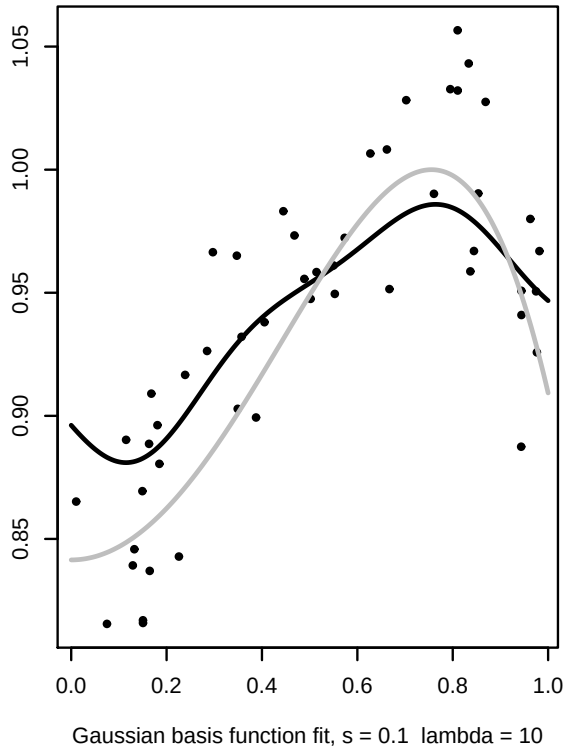


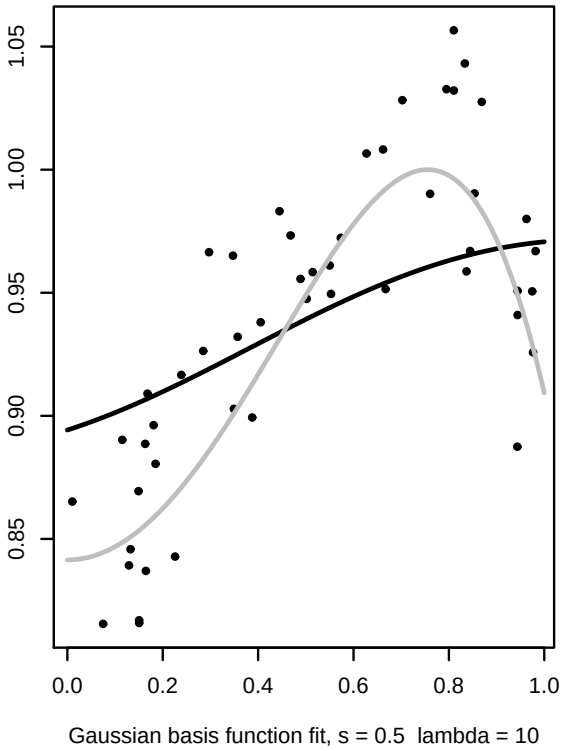


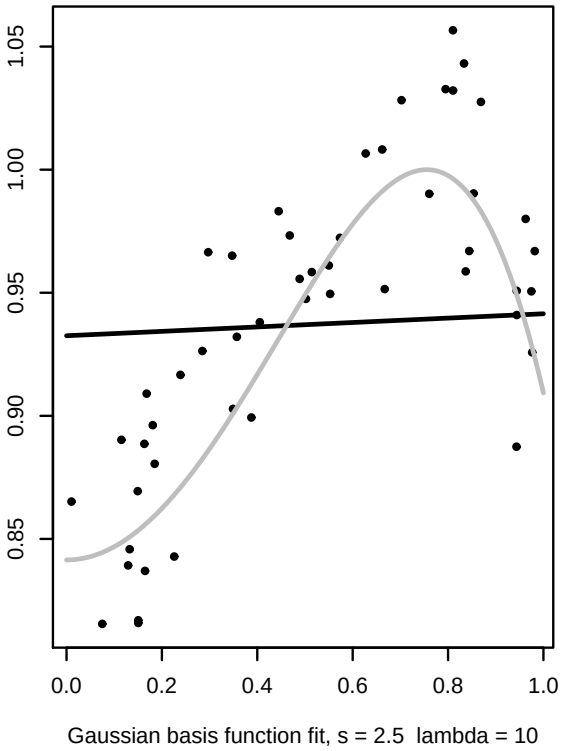


Yet More Results with Regularized Gaussian Basis Functions

Here are the results with $\lambda = 10$:

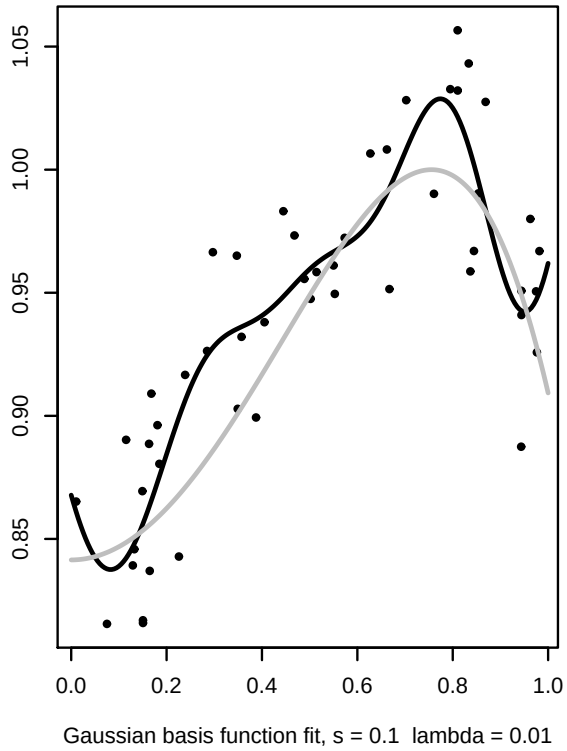




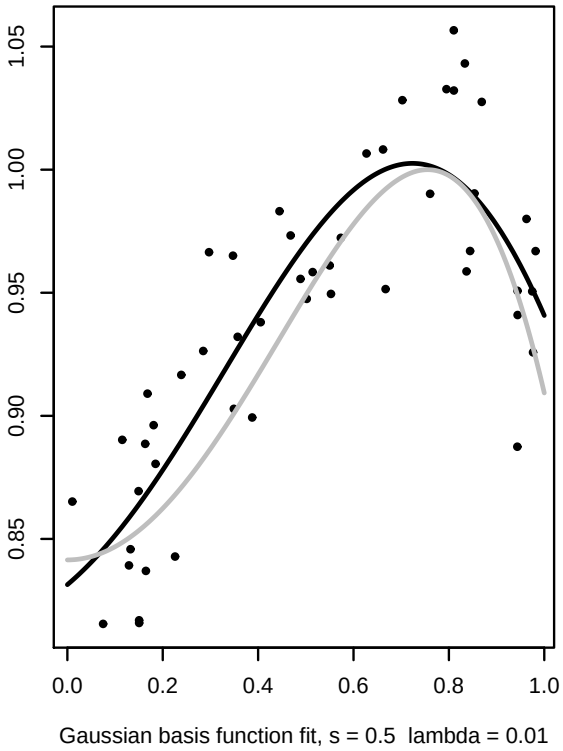


And Yet More Results...

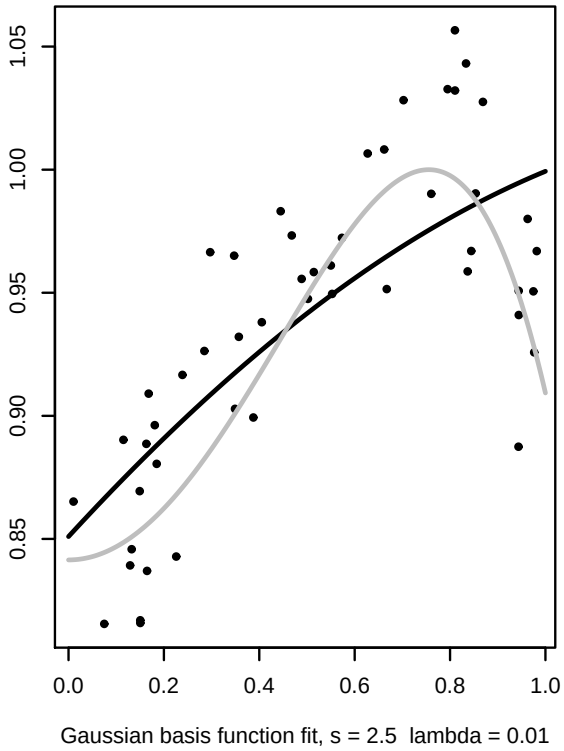
Here are the results with $\lambda = 0.01$:



Gaussian basis function fit, s = 0.1 lambda = 0.01



Gaussian basis function fit, s = 0.5 lambda = 0.01



Gaussian basis function fit, s = 2.5 lambda = 0.01

Conclusions from this Example

We see that we can control overfitting with Gaussian basis functions either by choosing the width of the basis functions, s, to be large, or by using a positive penalty, λ.

It seems that we may need to adjust *both* s and λ to get the best results.

One non-Bayesian approach is to set these by cross-validation.

We'll next look at Bayesian and semi-Bayesian approaches to these linear basis function models.