Radial Basis Function Networks

Introduction

In this lecture we will look at RBFs, networks where the activation of hidden is based on the **distance** between the input vector and a prototype vector

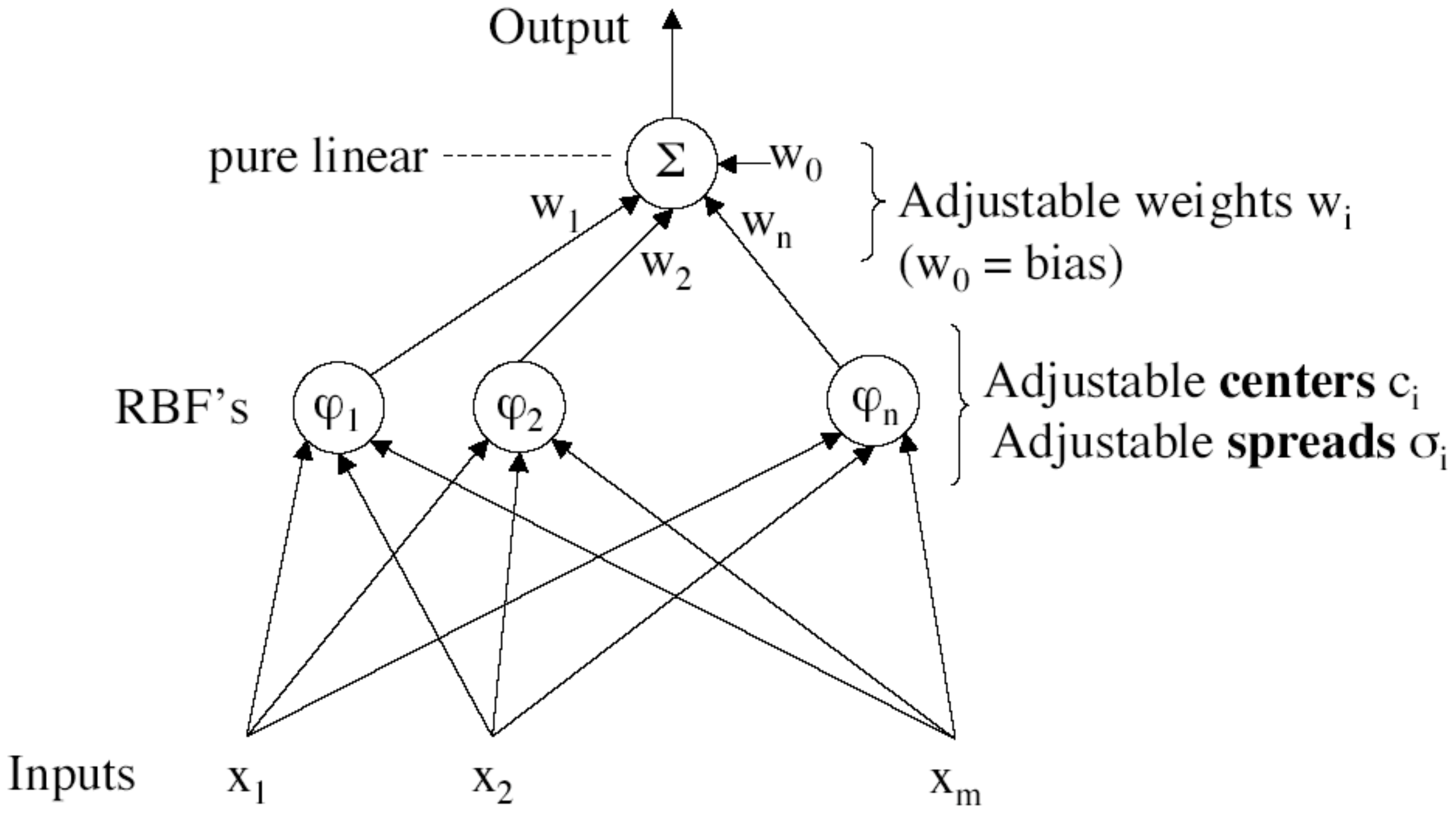
Radial Basis Functions have a number of interesting properties

- Strong connections to other disciplines
  - function approximation, regularization theory, density estimation and interpolation in the presence of noise [Bishop, 1995]
- RBFs allow for a straightforward interpretation of the internal representation produced by the hidden layer
- RBFs have training algorithms that are significantly faster than those for MLPs
- RBFs can be implemented as support vector machines

Radial Basis Functions

- Discriminant function flexibility
  - –NON-Linear
    - •But with sets of linear parameters at each layer
    - •Provably general function approximators for sufficient nodes
- Error function
  - –Mixed- Different error criteria typically used for hidden vs. output layers.
    - –Hidden: Input approx.
    - –Output: Training error
- Optimization
  - – Simple least squares - with hybrid training solution is unique.

Two layer Non-linear NN



Non-linear functions are radially symmetric 'kernels', with free parameters

Input to Hidden Mapping

- **Each region of feature space by means of a radially symmetric function**
  - Activation of a hidden unit is determined by the DISTANCE between the input vector *x* and a prototype vector *μ*

$$\varphi_j(x) = f\left(\left\|x - \mu_j\right\|\right)$$

- **Choice of radial basis function**
- Although several forms of radial basis may be used, Gaussian kernels are most commonly used
  - The Gaussian kernel may have a full-covariance structure, which requires D(D+3)/2 parameters to be learned

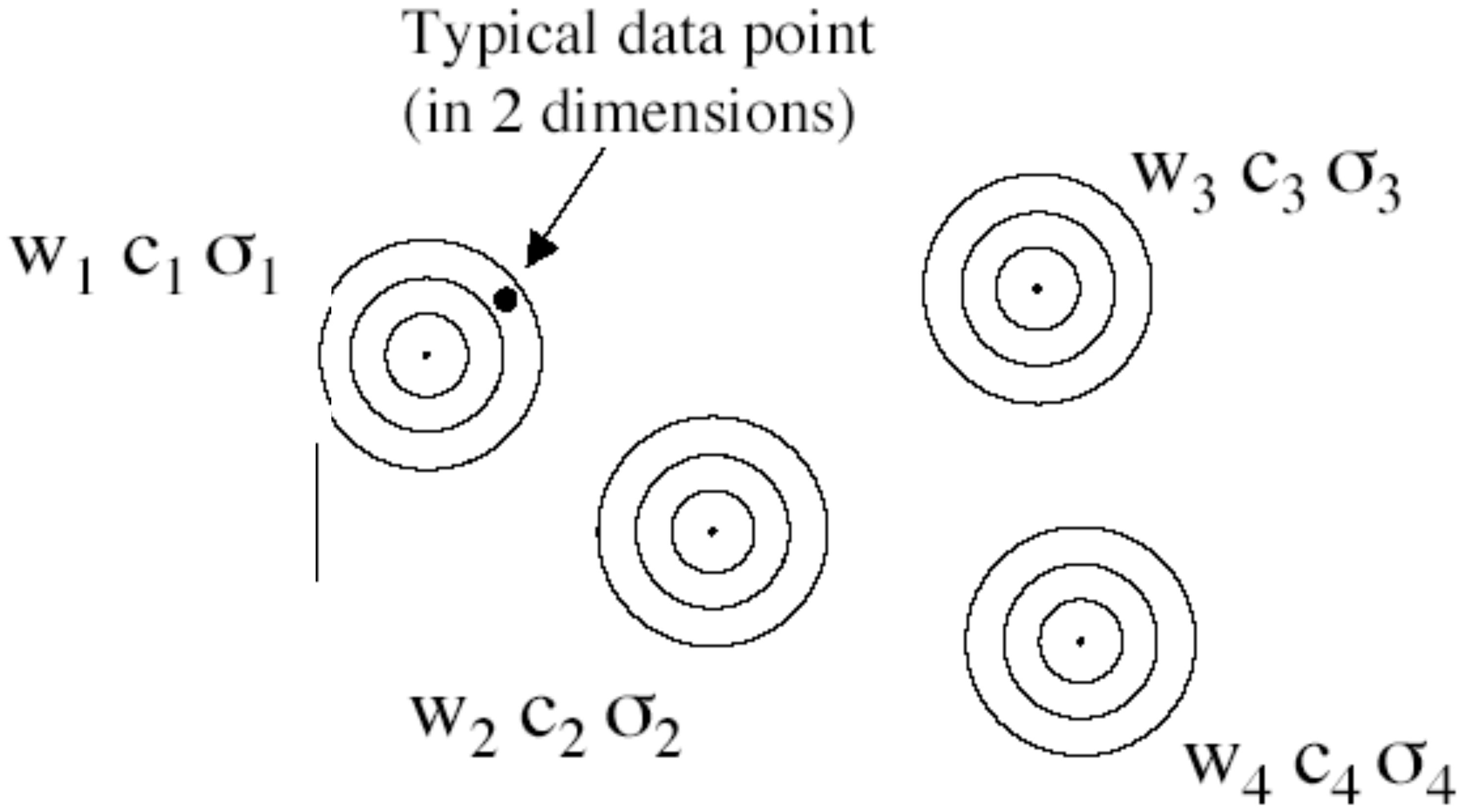
$$\varphi_j(x) = \exp\left[-\frac{1}{2}\left(x - \mu_j\right)^T \Sigma_j^{-1}\left(x - \mu_j\right)\right]$$

  - or a diagonal structure, with only (D+1) independent parameters

$$\varphi_j(x) = \exp\left[-\frac{\left\|x - \mu_j\right\|^2}{2\sigma_j^2}\right]$$

In practice, a trade-off exists between using a small number of basis with many parameters or a larger number of less flexible functions [Bishop, 1995]

Builds up a function out of 'Spheres'



RBF vs. NN

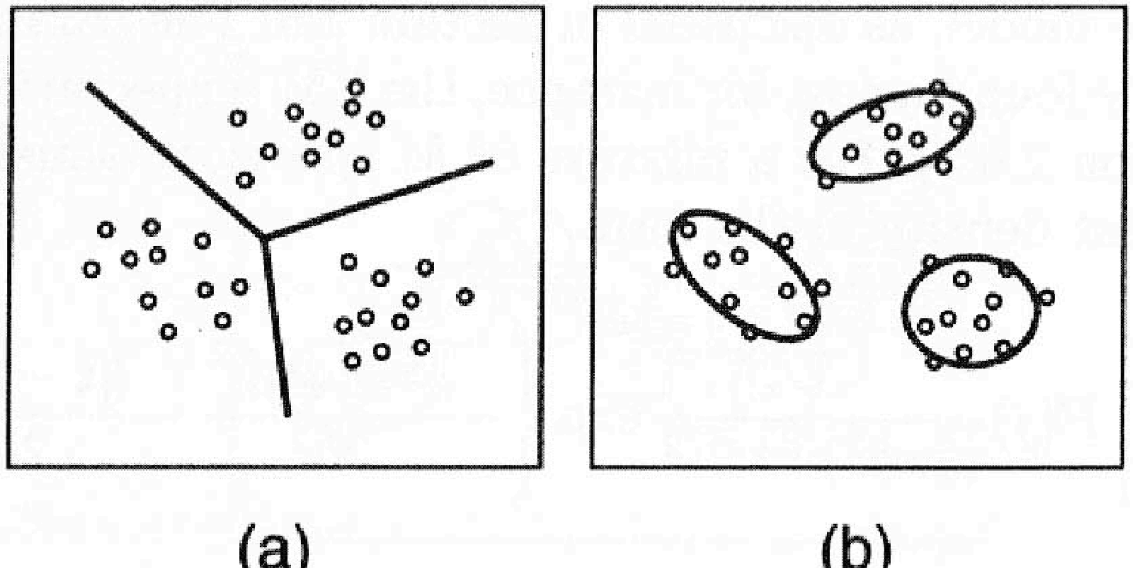
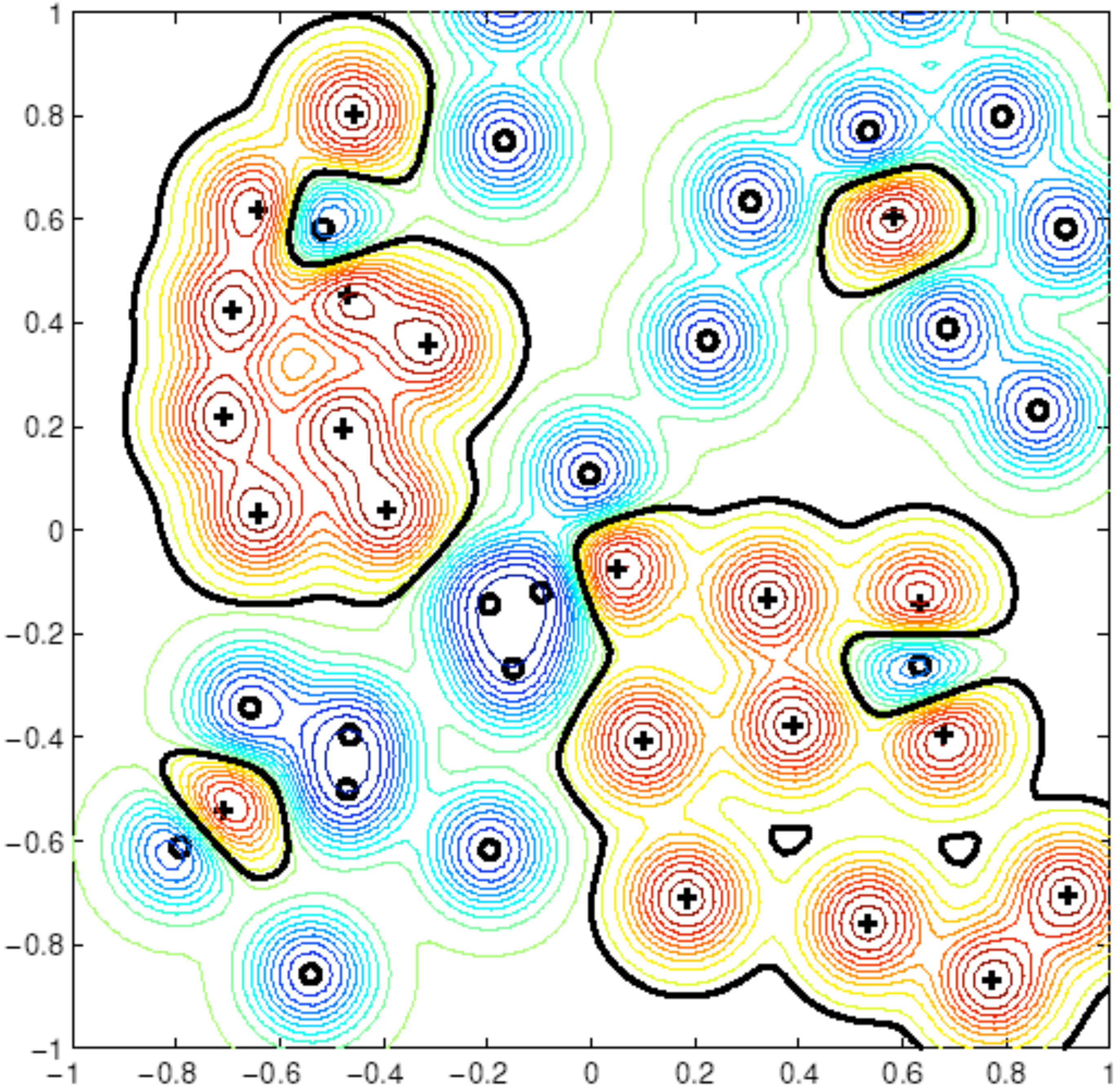
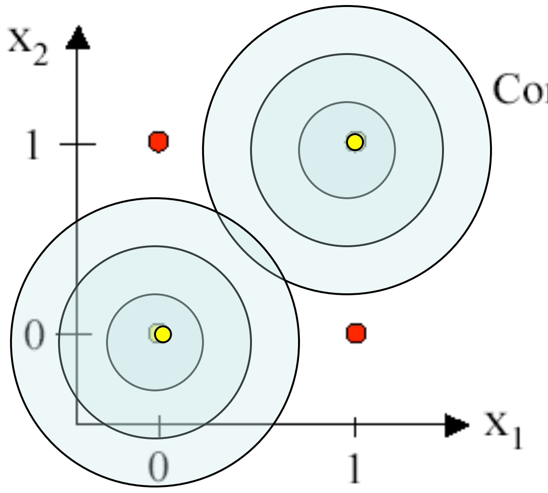


Figure 5.8. Schematic example of data points in two dimensions which fall into three distinct classes. One way to separate the classes is to use hyperplanes, shown in (a), as used in a multi-layer perceptron. An alternative approach, shown in (b), is to fit each class with a kernel function, which gives the type of representation formed by a radial basis function network.


1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1

XOR with RBF

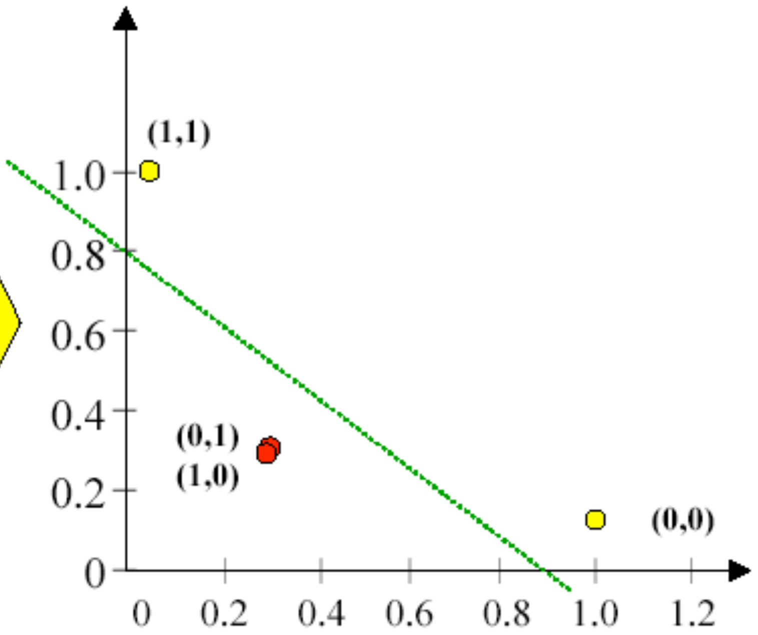


Consider the nonlinear functions to map the input vector **x** to the φ_1- φ_2 space

$$\mathbf{x}=[x_1\ x_2] \qquad \varphi_1(\mathbf{x}) = e^{-\|\mathbf{x}-\mathbf{t}_1\|^2} \qquad \mathbf{t}_1 = [1\ \ 1]^T$$

$$\varphi_2(\mathbf{x}) = e^{-\|\mathbf{x}-\mathbf{t}_2\|^2} \qquad \mathbf{t}_2 = [0\ \ 0]^T$$

Input x	$\varphi_1(x)$	$\varphi_2(x)$
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(1,0)	0.3678	0.3678
(0,0)	0.1353	1

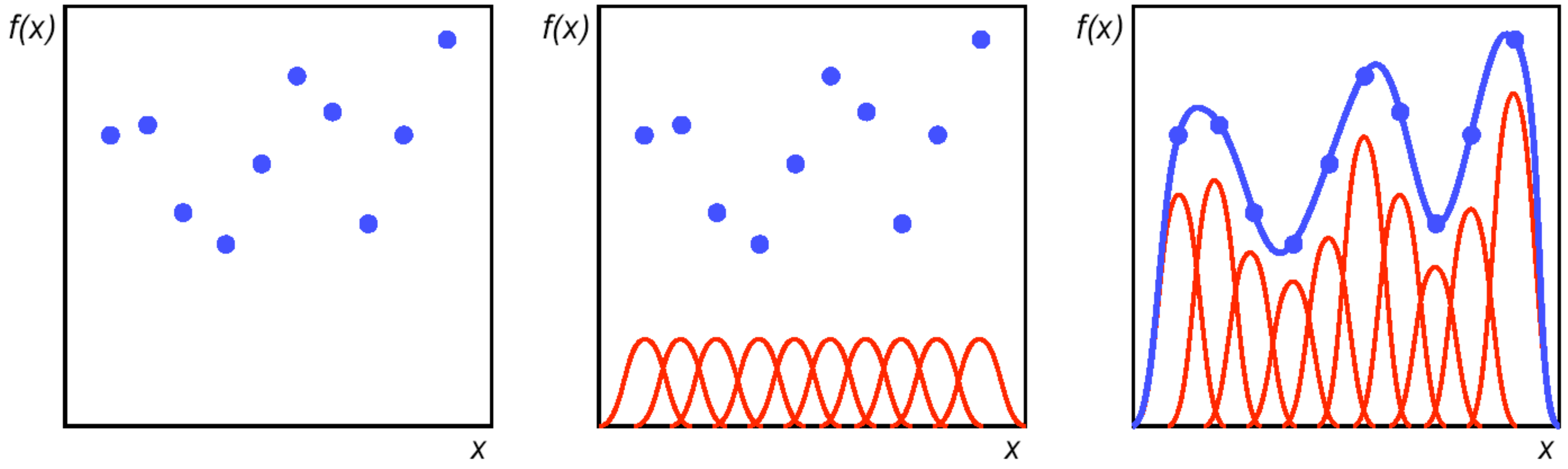


The nonlinear φ function transformed a nonlinearly separable problem into a linearly separable one !!!

RBFs have their origins in techniques for performing exact interpolation

These techniques place a basis function at each of the training examples *f(x)*

and compute the coefficients w_k so that the "mixture model" has zero error at examples



Formally: Exact Interpolation

Goal: Find a function $h(\mathbf{x})$, such that $h(\mathbf{x_i}) = \mathbf{t_i}$, for inputs $\mathbf{x_i}$ and targets $\mathbf{t_i}$

$$y = h(\mathbf{x}) = \sum_{\mathbf{i=1:n}} w_i \varphi\left(\left\|\mathbf{x} - \mathbf{x}_i\right\|\right)$$

Recall the Kernel trick

Form
$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1n} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \varphi_{n1} & \varphi_{n2} & \cdots & \varphi_{nn} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{bmatrix}$$

$$\varphi_{ij} = \varphi\left(\left\|\mathbf{x}_i - \mathbf{x}_j\right\|\right)$$

$$\mathbf{\Phi}\vec{\mathbf{w}} = \vec{\mathbf{t}}$$

Solve for w

$$\vec{\mathbf{w}} = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\vec{\mathbf{t}}$$

Solving conditions

- Micchelli's Theorem: If the points $\mathbf{x}_i$ are distinct, then the Φ matrix will be nonsingular.

Mhaskar and Micchelli, Approximation by superposition of sigmoidal and radial basis functions, Advances in Applied Mathematics,13, 350-373, 1992.

Radial Basis Function Kernel

- Radial basis function kernel

$$k(\mathbf{x}, \mathbf{z}) = \exp\left[-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{z}\|^2\right]$$

  Here σ controls the "width" of the kernel.

- Kernel only depends on the difference $\mathbf{x} - \mathbf{z}$, i.e. the relative location of the points to one another, = shift invariance

- Kernel only depends on the Euclidean distance between two patterns, $k(\mathbf{x}, \mathbf{z}) = \tilde{k}(\|\mathbf{x} - \mathbf{z}\|)$, = radially symmetric

Radial Basis function Networks

n outputs, m basis functions

$$y_\mathrm{k} = \mathrm{h}_\mathrm{k}(\mathbf{x}) = \sum_{i=1:m} w_{ki}\varphi_i(\mathbf{x}) + w_{k0}, \qquad \text{e.g. } \varphi_i(\mathbf{x}) = \exp\left(-\frac{\left\|\mathbf{x}-\mu_j\right\|^2}{2\sigma_j^2}\right)$$

$$= \sum_{i=0:m} w_{ki}\varphi_i(\mathbf{x}), \qquad \varphi_0(\mathbf{x}) = 1$$

Rewrite this as:

$$\vec{\mathbf{y}} = \mathbf{W}\vec{\varphi}(\mathbf{x}), \quad \vec{\varphi}(\mathbf{x}) = \begin{bmatrix} \varphi_0(\mathbf{x}) \\ \vdots \\ \varphi_m(\mathbf{x}) \end{bmatrix}$$

Learning

RBFs are commonly trained following a hybrid procedure that operates in two stages

Unsupervised selection of RBF centers

- RBF centers are selected so as to match the distribution of training examples in the input feature space
- This is the critical step in training, normally performed in a slow iterative manner
- A number of strategies are used to solve this problem

- Supervised computation of output vectors
  - Hidden-to-output weight vectors are determined so as to minimize the sum-squared
  - error between the RBF outputs and the desired targets
  - Since the outputs are linear, the optimal weights can be computed using fast, linear

Least squares solution for output weights

Given $\boldsymbol{L}$ input vectors x_l, with labels t_l

Form the least square error function:

$$E = \sum_{l=1:L} \sum_{k=1:n} \left\{ y_k(\mathbf{x}_l) - t_l^k \right\}^2, \quad t_l^k \text{ is the } k^{\text{th}} \text{ value of the target vector}$$

$$E = \sum_{l=1:L} (\vec{\mathbf{y}}(\mathbf{x}_l) - \vec{\mathbf{t}}_l)^t (\vec{\mathbf{y}}(\mathbf{x}_l) - \vec{\mathbf{t}}_l)$$

$$= \sum_{l=1:L} (\vec{\varphi}(\mathbf{x}_l)\mathbf{W}^t - \vec{\mathbf{t}}_l)^t (\vec{\varphi}(\mathbf{x}_l)\mathbf{W}^t - \vec{\mathbf{t}}_l)$$

$$E = \sum_{l=1:L} (\mathbf{\Phi}\mathbf{W}^t - \mathbf{T})^t (\mathbf{\Phi}\mathbf{W}^t - \mathbf{T}) = \sum_{l=1:L} \mathbf{W}\mathbf{\Phi}^t\mathbf{\Phi}\mathbf{W}^t - 2\mathbf{W}\mathbf{\Phi}^t\mathbf{T} + \mathbf{T}^t\mathbf{T}$$

$$\frac{\partial E}{d\mathbf{W}} = 0 = 2\mathbf{\Phi}^t\mathbf{\Phi}\mathbf{W}^t - 2\mathbf{\Phi}^t\mathbf{T}$$

$$\mathbf{W}^t = \left(\mathbf{\Phi}^t\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^t\mathbf{T}$$

Solving for input parameters

- Now we have a linear solution, if we know the input parameters. How do we set the input parameters?
  - Gradient descent--like Back-prop
  - Density estimation viewpoint: By looking at the meaning of the RBF network for classification, the input parameters can be estimated via density estimation and/or clustering techniques
  - We will skip these in lecture

Unsupervised methods overview

- **Random selection of centers**
  - The simplest approach is to randomly select a number of training examples as RBF centers
  - This method has the advantage of being very fast, but the network will likely require an excessive number of centers
  - Once the center positions have been selected, the spread parameters σ can be estimated, for instance, from the average distance between neighboring centers
- **Clustering**
  - Alternatively, RBF centers may be obtained with a clustering procedure such as the k-means algorithm

  The spread parameters can be computed as before, or from the sample covariance of the examples of each cluster
- **Density estimation**
  - The position of the RB centers may also be obtained by modeling the feature space density with a Gaussian Mixture Model using the Expectation- Maximization algorithm
  - The spread parameters for each center are automatically obtained from the covariance matrices of the corresponding Gaussian components

Probabilistic Interpretion of RBFs for Classification

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{p(\mathbf{x})}$$

$$= \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{\sum_{k'} p(\mathbf{x}|\mathcal{C}_{k'})P(\mathcal{C}_{k'})}. \qquad = \sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x})$$

$$\phi_k(\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)}{\sum_{k'} p(\mathbf{x}|\mathcal{C}_{k'})P(\mathcal{C}_{k'})}$$

Introduce Mixture

$$p(\mathbf{x}|\mathcal{C}_k) = \sum_{j=1}^{M} p(\mathbf{x}|j)P(j|\mathcal{C}_k).$$

Prob RBF con'd

$$P(C_k|\mathbf{x}) = \frac{\sum_{j=1}^{M} P(j|C_k)p(\mathbf{x}|j)P(C_k)}{\sum_{j'=1}^{M} p(\mathbf{x}|j')P(j')} \frac{P(j)}{P(j)}$$

$$\boxed{= \sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x})}$$

where

$$P(j) = \sum_k P(j|C_k)P(C_k).$$

<u>**Basis function outputs**</u>: Posterior *j*th Feature probabilities

<u>**weights**</u>: Class probability given *j*th Feature value

$$\phi_j(\mathbf{x}) = \frac{p(\mathbf{x}|j)P(j)}{\sum_{j'=1}^{M} p(\mathbf{x}|j')P(j')}$$

$$= P(j|\mathbf{x})$$

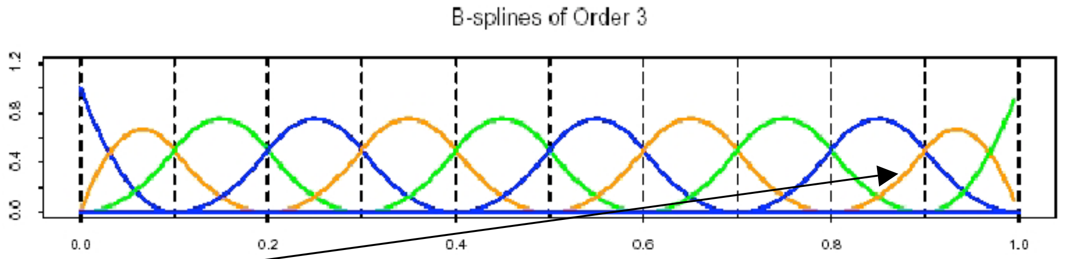
$$w_{kj} = \frac{P(j|C_k)P(C_k)}{P(j)}$$

$$= P(C_k|j).$$

Regularized Spline Fits

Given

$$\Phi = \begin{bmatrix} \phi_1(X_1) & \cdots & \phi_N(X_1) \\ & \vdots & \\ \phi_1(X_M) & \cdots & \phi_N(X_M) \end{bmatrix}$$



$\mathrm{y}_{\mathrm{pred}} = \Phi w$

Minimize

$L = (\mathrm{y} - \Phi w)^T (\mathrm{y} - \Phi w) + \lambda w^T \Omega w$

Make a penalty on large curvature

$$\Omega_{ij} = \int \left(\frac{d^2\phi_i(x)}{dx^2} \right)\left(\frac{d^2\phi_j(x)}{dx^2} \right) dx$$

Minimize

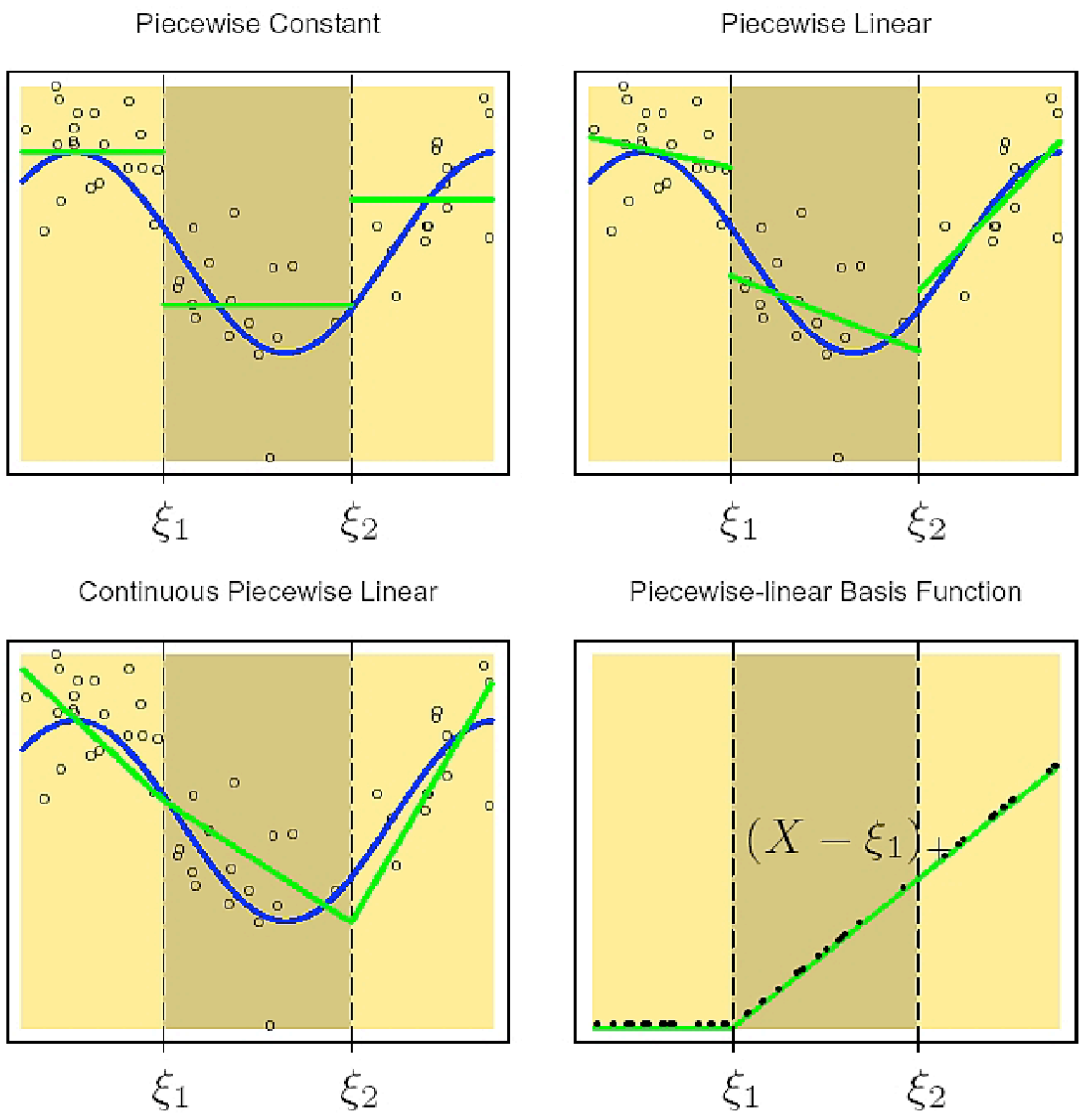
$$L = (\mathrm{y} - \Phi w)^T (\mathrm{y} - \Phi w) + \lambda w^T \Omega w$$

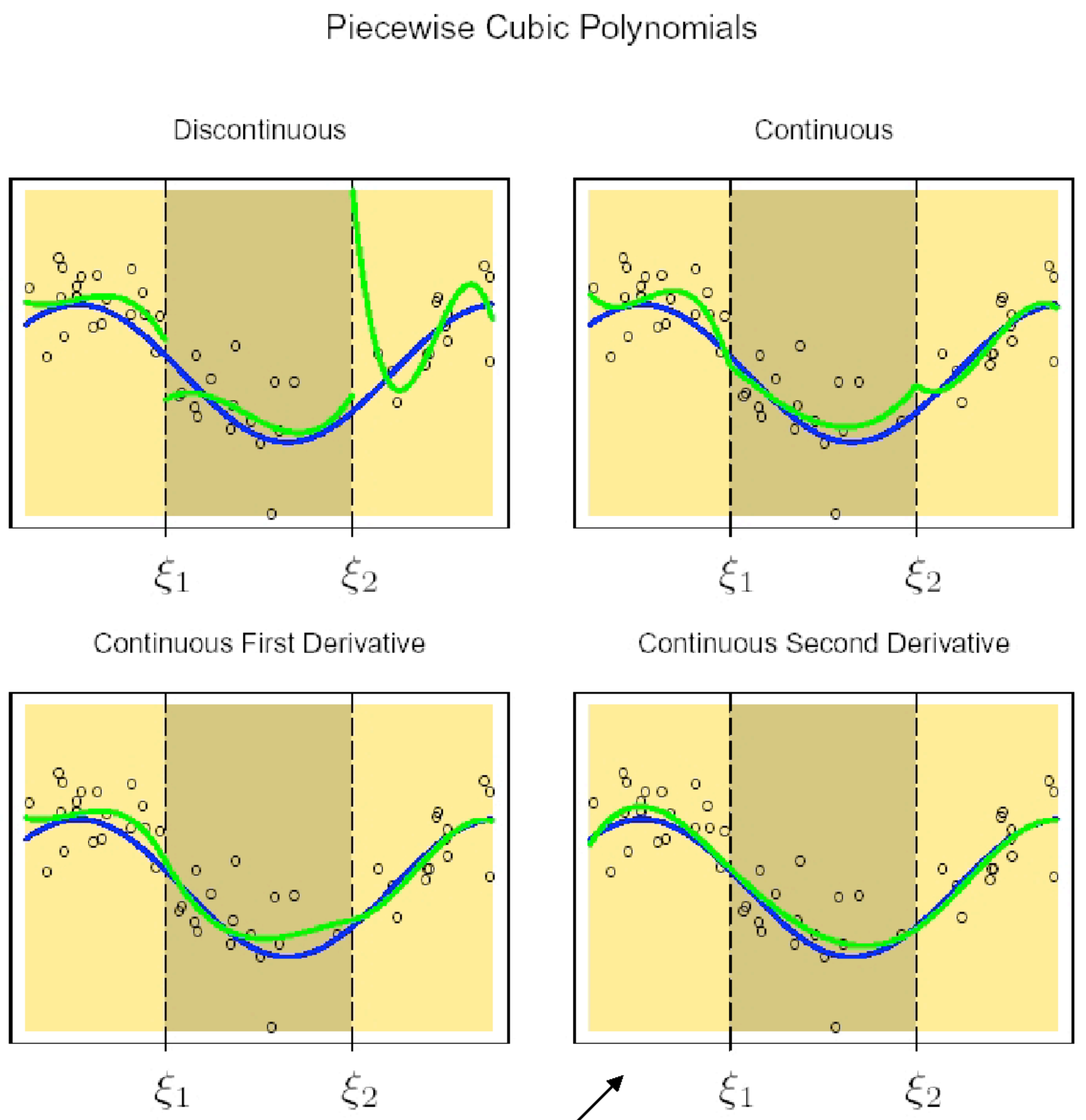
Solution

$$w = \left(\Phi^T \Phi + \lambda \Omega\right)^{-1} \Phi^T \vec{y}$$

Generalized Ridge Regression

Piecewise Polynomials and Splines





Cubic Spline Truncated Power Basis

$$1,\ X,\ X^2,\ X^3,\ (X-\xi_1)^3_+,\ (X-\xi_2)^3_+$$

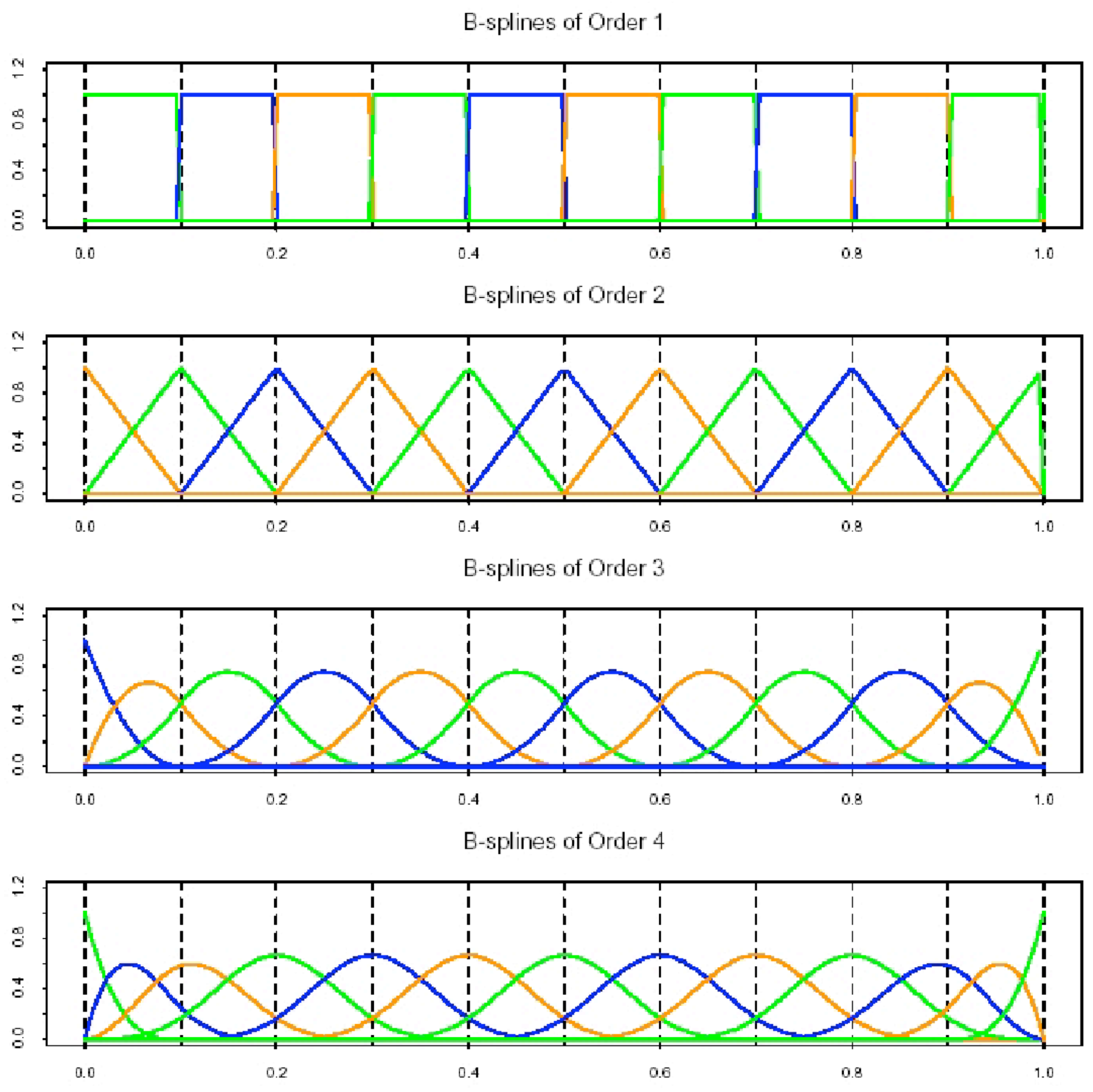


Figure 5.17: *The sequence of B-splines up to order four with ten knots evenly spaced from 0 to 1. The B-splines have* local support*; they are nonzero on an interval spanned by $M + 1$ knots.*

Equivalent Kernel

- Linear Regression solution admit a kernel interpretation

$$w = \left(\Phi^T\Phi + \lambda\Omega\right)^{-1}\Phi^T\vec{y}$$

$$y_{pred}(x) = \Phi(x)w = \Phi(x)\left(\Phi^T\Phi + \lambda\Omega\right)^{-1}\Phi^T\vec{y}$$

$$Let \quad \left(\Phi^T\Phi + \lambda\Omega\right)^{-1}\Phi^T = S_\lambda\Phi^T$$

$$y_{pred}(x) = \Phi(x)\sum S_\lambda\vec{\phi}(x_i)y_i$$

$$= \sum \Phi(x)S_\lambda\vec{\phi}(x_i)y_i$$

$$= \sum K(x, x_i)y_i$$

Eigenanalysis and DF

- An eigenanalysis of the effective kernel gives the effective degrees of freedom (DF), i.e. # of basis functions

Evaluate K at all points

$$\mathbf{K}_{ij} = K(x_i, x_j)$$

Eigenanalysis

$$\mathbf{K} = VDV^{-1} = USU^T$$

Equivalent Basis Functions

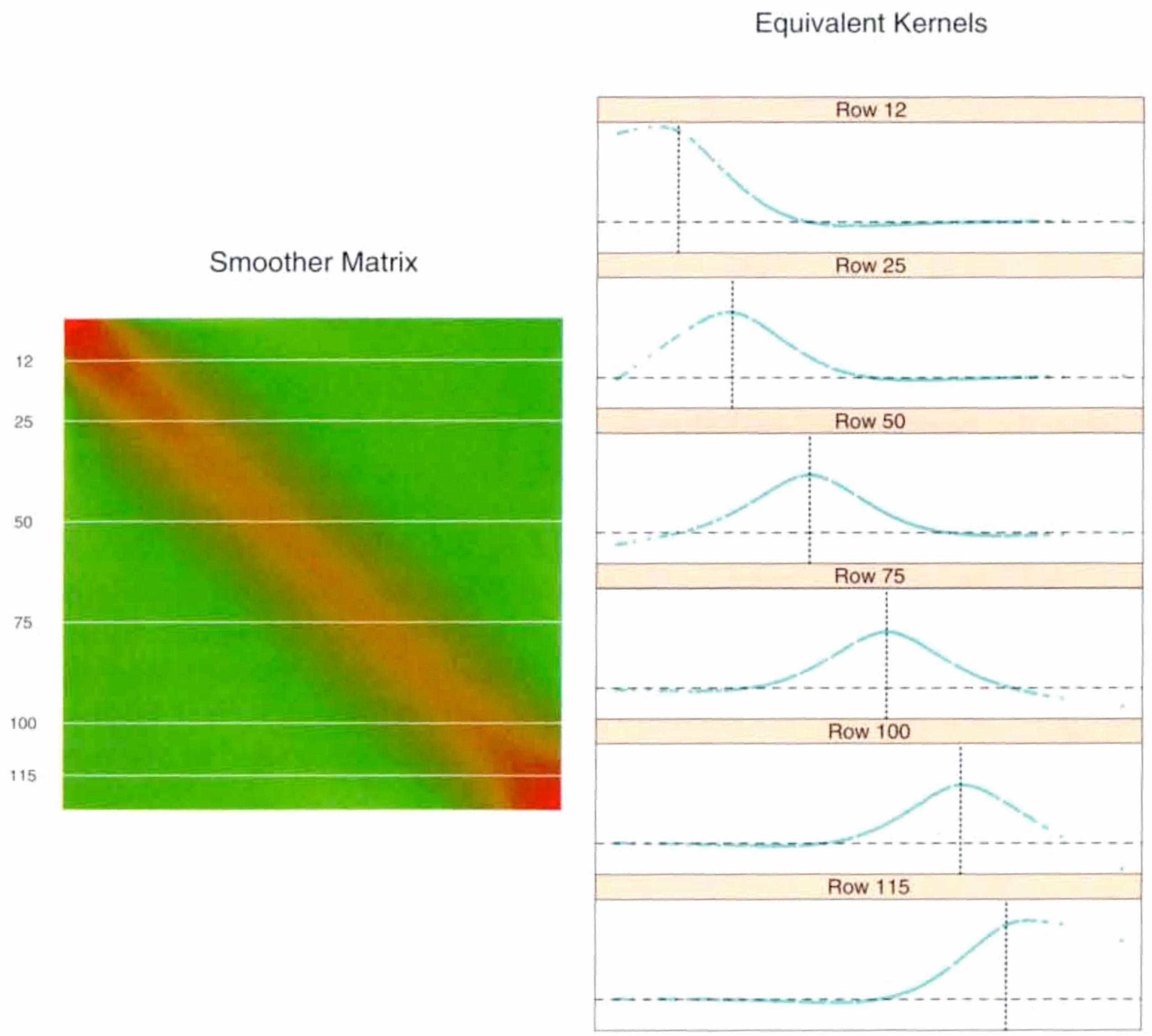


FIGURE 5.8. *The smoother matrix for a smoothing spline is nearly banded, indicating an equivalent kernel with local support. The left panel represents the elements of* **S** *as an image. The right panel shows the equivalent kernel or weighting function in detail for the indicated rows.*

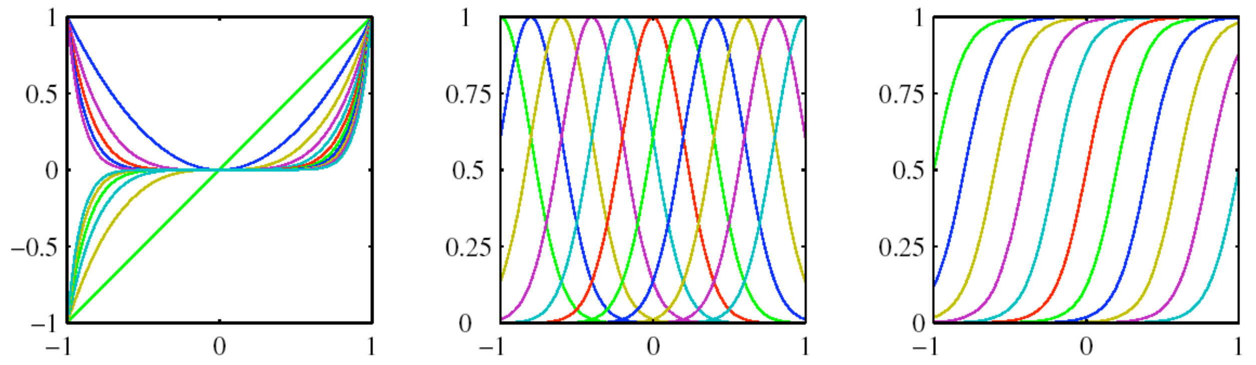


Figure 3.1 Examples of basis functions, showing polynomials on the left, Gaussians of the form (3.4) in the centre, and sigmoidal of the form (3.5) on the right.

Figure 3.10 The equivalent kernel $k(x, x')$ for the Gaussian basis functions in Figure 3.1, shown as a plot of x versus x', together with three slices through this matrix corresponding to three different values of x. The data set used to generate this kernel comprised 200 values of x equally spaced over the interval $(-1, 1)$.

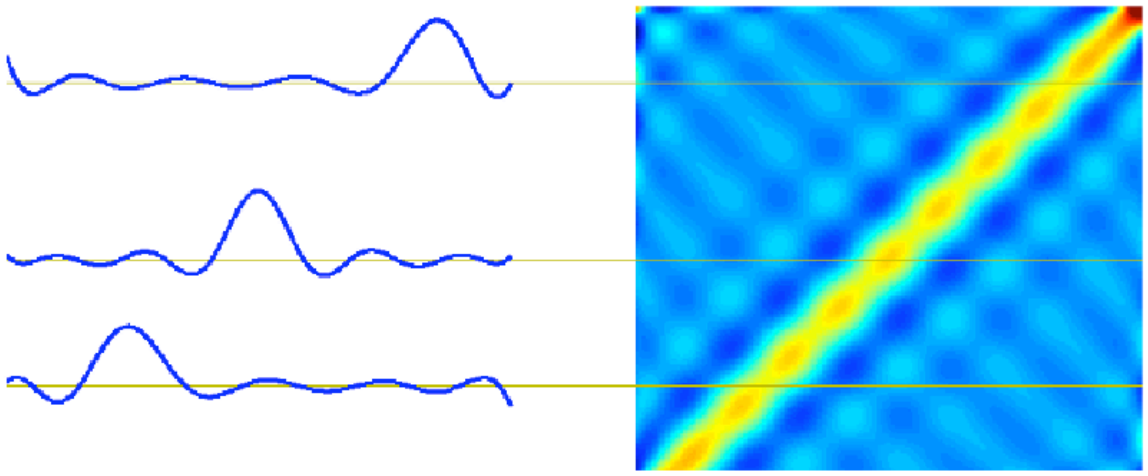
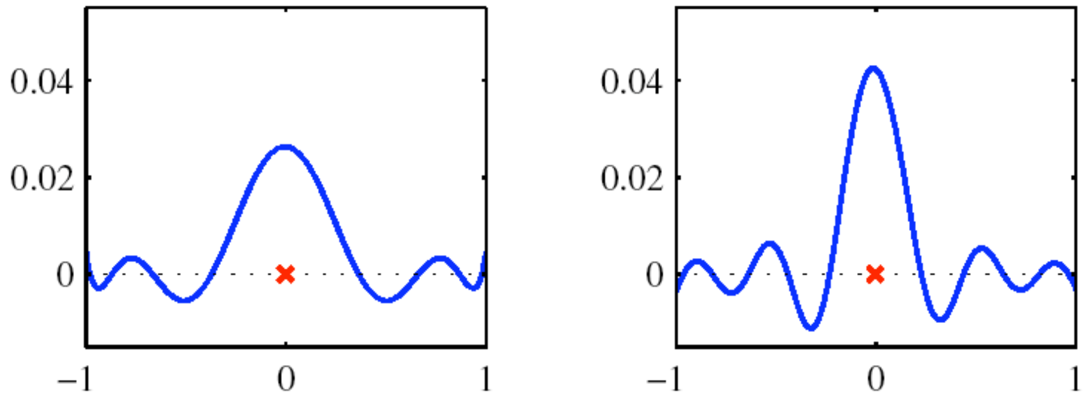


Figure 3.11 Examples of equivalent kernels $k(x, x')$ for $x = 0$ plotted as a function of x', corresponding (left) to the polynomial basis functions and (right) to the sigmoidal basis functions shown in Figure 3.1. Note that these are localized functions of x' even though the corresponding basis functions are nonlocal.



Computing Error Bars on Predictions

- It is important to be able to compute error bars on linear regression predictions. We can do that by propagating the error in the measured values to the predicted values.

Given the Gram Matrix

$$\Phi = \begin{bmatrix} \phi_1(X_1) & \cdots & \phi_N(X_1) \\ & \vdots & \\ \phi_1(X_M) & \cdots & \phi_N(X_M) \end{bmatrix}$$

Simple Least squares

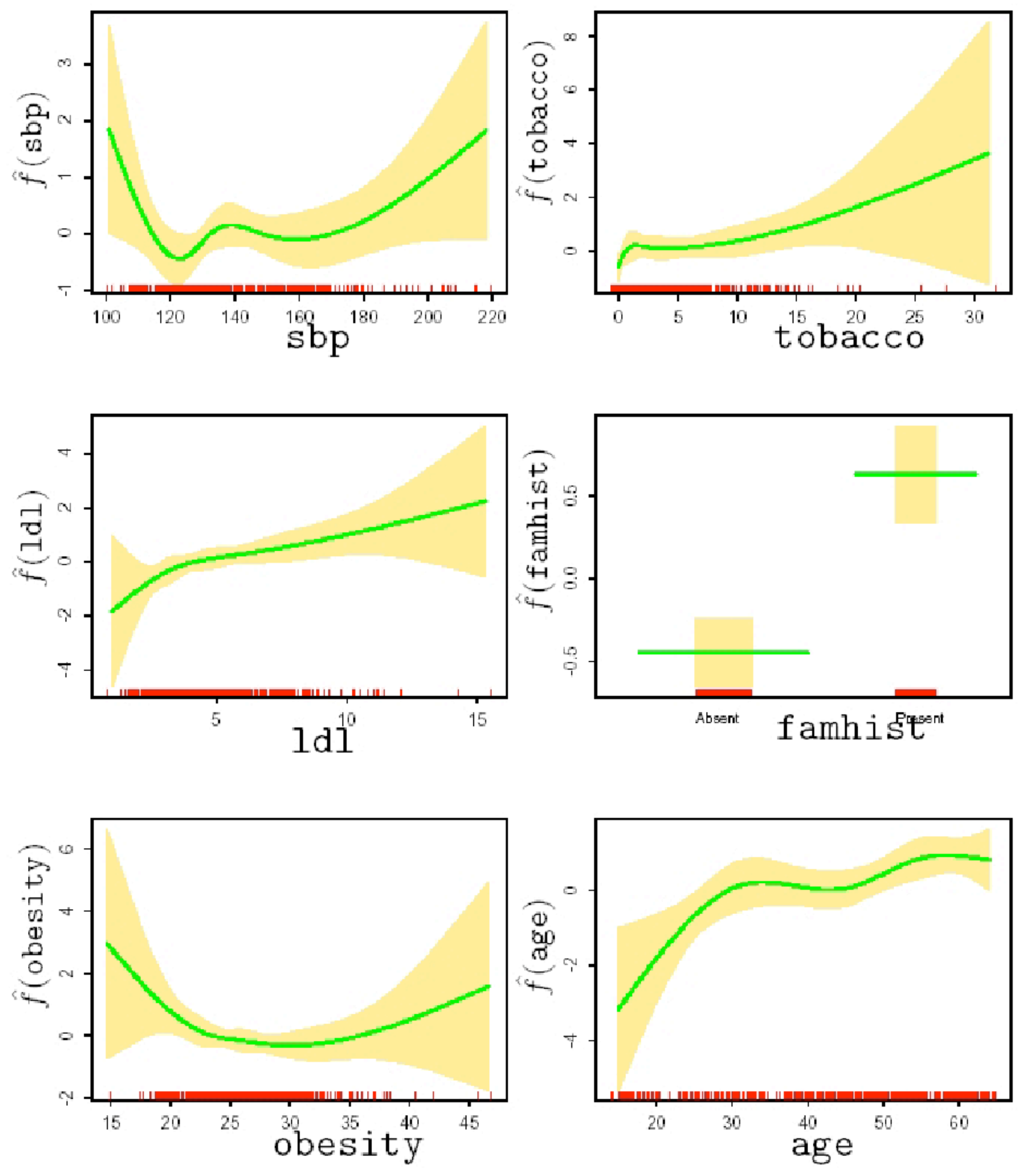
$$w = \left(\Phi^T \Phi\right)^{-1} \Phi^T \vec{y}$$

predictions $\quad y_{pred} = \Phi w = \Phi\left(\Phi^T \Phi\right)^{-1} \Phi^T \vec{y} = S\vec{y}$

Error $\quad \operatorname{cov}[y_{pred}] = \operatorname{cov}[S\vec{y}] = S \operatorname{cov}[\vec{y}] S^T = S(\beta \mathbf{I}) S^T = \beta S S^T$

Fitted Natural Spline Model

- Each term 4 df (except famhist)
- Pointwise standard error curves
- Rugplot shows location of x values



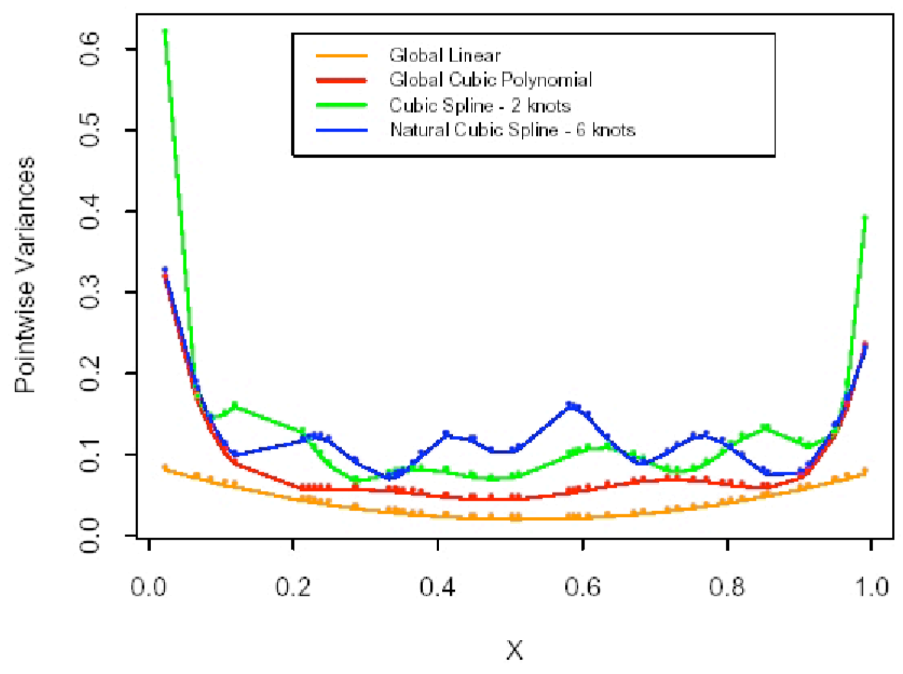


Figure 5.3: *Pointwise variance curves for four different models, with X consisting of 50 points drawn at random from $U[0,1]$, and an assumed error model with constant variance. The linear and cubic polynomial fits have two and four degrees of freedom respectively, while the cubic spline and natural cubic spline each have six degrees of freedom. The cubic spline has two knots at 0.33 and 0.66, while the natural spline has boundary knots at 0.1 and 0.9, and four interior knots uniformly spaced between them.*

With $y_i = f(x_i) + \epsilon_i$

$\epsilon_i \sim$ iid $(0, \sigma^2)$

var$\hat{f}(x) = h(x)^T (H^T H)^{-1} h(x) \sigma^2$

(training data assumed fixed)

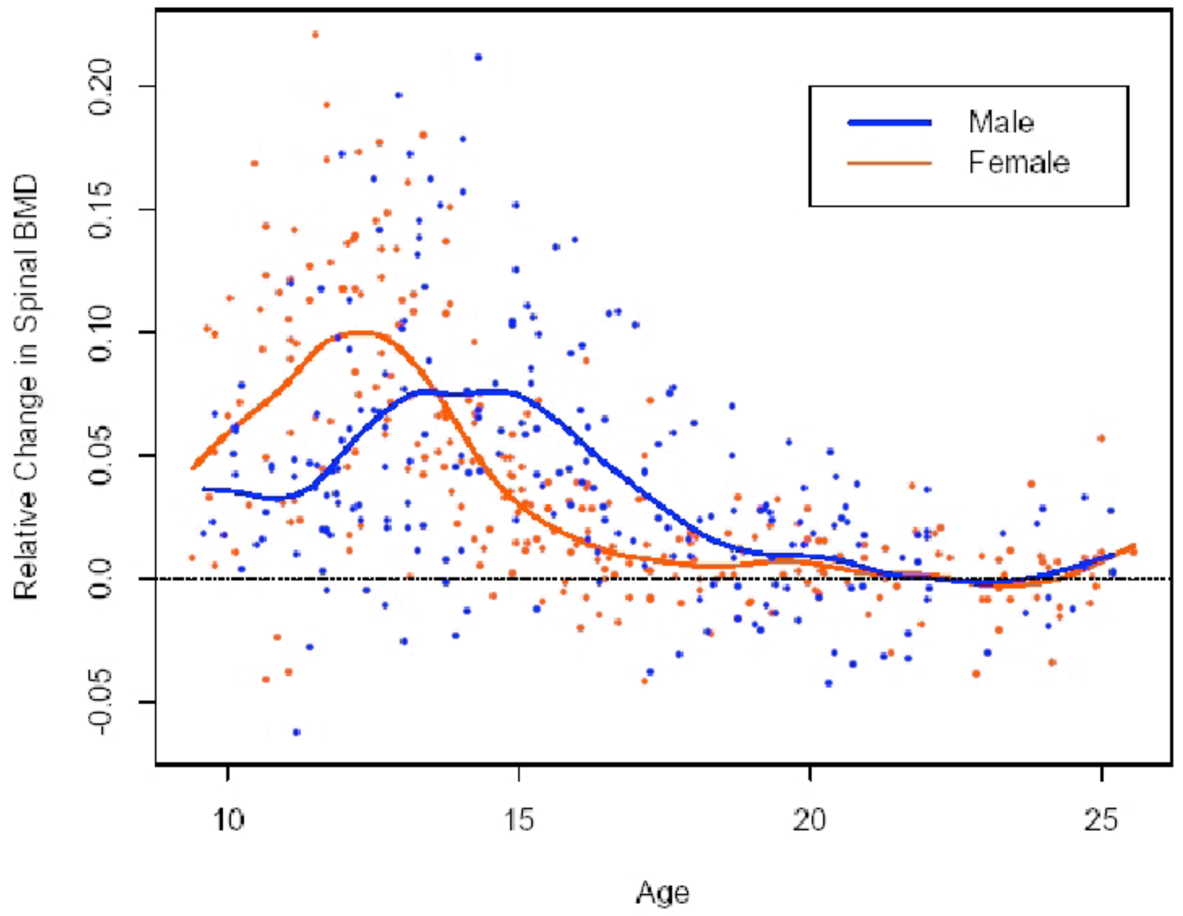


Figure 5.6: *The response is the relative change in bone mineral density measured at the spine in adolescents, as a function of age. A separate smoothing spline was fit to the males and females, with* $\lambda \approx 0.00022$*. This choice corresponds to about* 12 *degrees of freedom.*