

Lecture 5: Beyond Linearity: Basis Expansions, Local Averaging & Kernel Smoothing

Readings: ESL (Ch. 5.1–5.7, Ch. 6), ISL (Ch. 7), Bach (Ch. 6); code

Topics: going beyond linearity with basis expansions, global models built from local pieces: splines, local methods that refit everywhere: local averaging & kernel smoothing, kernel density estimation and classification, appendix

1 Going Beyond Linearity With Basis Expansions

1.1 Moving Beyond Linearity

Several ways to extend linear models to increase model flexibility:

- **Polynomial Regression (Lecture 2):** Add additional predictors which are powers of the original ones (e.g., X, X^2, X^3).
- **Step Functions:** Fit a piecewise constant function by cutting the range of a predictor into distinct regions, treating them as a qualitative variable.
- **Regression Splines:** Divide the range of a predictor into several regions and fit a polynomial in each. These polynomials are constrained to join smoothly at the region boundaries (knots).
- **Smoothing Splines:** Similar to regression splines but are derived by minimizing an RSS criterion that includes a penalty term to control the smoothness of the function.
- **Local Regression:** Compute the regression fit locally. The regions are allowed to overlap smoothly, and the fit at a point is influenced most by the data points nearest to it.
- **Generalized Additive Models (GAMs) (later lecture):** Extend the above methods to scenarios with multiple predictors, modeling the response as a sum of smooth functions of each predictor.

1.2 Moving Beyond Linearity: The Feature Map View

💡 General idea: Use a *pre-specified*¹ set of transformations on the inputs so that linear models work well on the transformed inputs (**features**).

Find some **feature map** $\phi : \mathbb{R}^d \rightarrow \mathcal{H} = \mathbb{R}^D$ (D possibly bigger than d) such that a linear model

$$f(X) = \beta^T \phi(X)$$

will be a good model for the transformed data $\{(\phi(x_i), y_i)\}_{i=1}^N$. Here, $\beta \in \mathbb{R}^D$ is the model parameter, the $\phi(x_i) \in \mathbb{R}^D$ are the features, and the $x_i \in \mathbb{R}^d$ are the original input samples.

🔴 What are some nice examples of features and feature maps?

¹In later lectures we will look at neural networks, which use adjustable transformations and learn their parameters to approximate the target function.

1.3 The Core Idea of Basis Expansions

Make linear models more flexible by augmenting/replacing the inputs X with a *pre-specified* set of transformations, or **basis functions**, $h_m(X) : \mathbb{R}^d \rightarrow \mathbb{R}$.

Definition 1.1 (Linear Basis Model). The model takes the general form:

$$f(X) = \sum_{m=1}^M \beta_m h_m(X), \quad (1)$$

where h_m is a basis function / feature map.

- The function $f(X)$ is now a flexible, nonlinear function of the input X .
- The model is still a **linear** model with respect to the coefficients β_m .
- The ERM problem is easy: we can use all the machinery of linear models (OLS, ridge, or lasso) on the new “predictor” matrix \mathbf{H} , with $\mathbf{H}_{im} = h_m(x_i)$.

This is a **global method**: the value of the prediction function at any point x depends on all of the β_m , which in turn depend on all of the training data.

1.4 Piecewise Polynomials

Assume $X \in \mathbb{R}$ ($d = 1$) from now on for simplicity. Polynomials are restricted by their global nature. Local representations are more useful.

Definition 1.2 (Piecewise Polynomial). Divide the domain of X into contiguous intervals and represent f by a separate polynomial in each interval.

The simplest case is piecewise constants (step functions):

$$f(X) = \sum_{m=1}^M c_m I(X \in C_m),$$

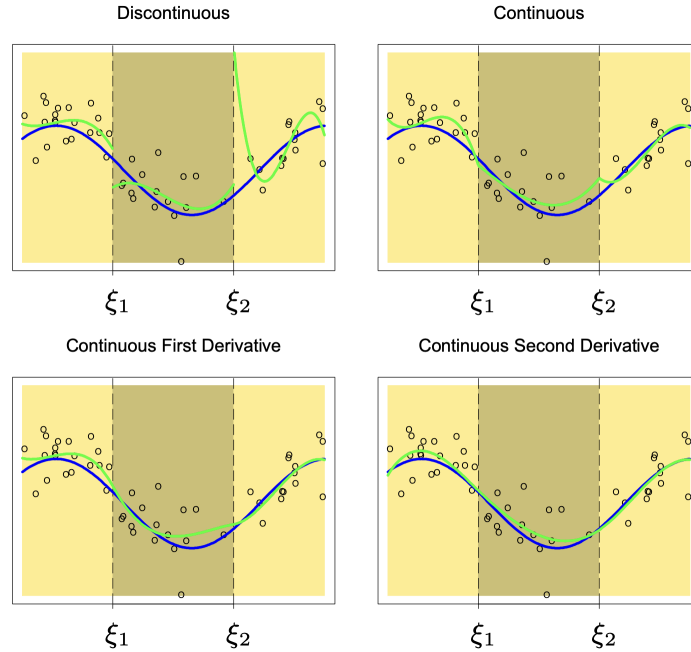
where C_m are disjoint regions covering \mathbb{R} .

- Unless constrained to be continuous, they can jump at the boundaries.
- Discontinuities may not be desirable for regression.
- We need to address smoothness at the knots.

💡 **Solution:** Impose continuity constraints at the boundary points (knots) ξ_j .

1.5 Why Smoother Functions are Preferred

A series of piecewise cubic polynomials fit to data, with increasing orders of continuity (see ESL Ch. 5.2):



2 Global Models Built From Local Pieces: Splines

2.1 Splines

💡 A direct solution: build a basis that incorporates the desired constraints.

Definition 2.1 (Order- M Spline). An order- M spline with knots $\xi_j, j = 1, \dots, K$ is a piecewise polynomial of order M (degree $M - 1$), and has continuous derivatives up to order $M - 2$.

A cubic spline (order 4) has continuous first and second derivatives at each knot.

Proposition 2.2 (Cubic Regression Spline with One Knot). A function $f : \mathbb{R} \rightarrow \mathbb{R}$ of the form

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - \xi)_+^3, \quad (2)$$

is a cubic spline with a knot at ξ . Here,

$$(x - \xi)_+^3 = \begin{cases} (x - \xi)^3, & x > \xi, \\ 0, & x \leq \xi, \end{cases}$$

and the $\beta_i \in \mathbb{R}$ are any coefficients.

Proof. See blackboard. □

2.2 Cubic Splines with K Knots in the Truncated Power Basis

In fact, the set

$$\{1, x, x^2, x^3, (x - \xi)_+^3\}$$

forms a basis for the vector space of all cubic splines with a single knot at ξ . Thus, any such cubic spline function can be uniquely represented as a linear combination of these basis functions.

In general, a cubic spline with K knots can be represented by a basis of $K + 4$ functions:

$$h_1(X) = 1, \quad h_2(X) = X, \quad h_3(X) = X^2, \quad h_4(X) = X^3,$$

$$h_{j+4}(X) = (X - \xi_j)_+^3, \quad j = 1, \dots, K,$$

where $(z)_+ = \max(0, z)$ is the positive part function.

💡 The truncated power functions $(X - \xi_j)_+^3$ are zero to the left of ξ_j and behave like $(X - \xi_j)^3$ to the right, allowing for smooth joins.

🔴 For $K = 2$, show that the truncated power basis functions represent a basis for a cubic spline with two knots.

2.3 Natural Cubic Splines

Definition 2.3 (Natural Cubic Spline). A natural cubic spline adds the constraint that the function is linear beyond the boundary knots.

💡 The behavior of polynomials fit to data is notoriously bad at the boundaries. The linear constraint prevents wild extrapolation.

Starting from the truncated power basis and imposing the boundary constraints

$$f''(x) = 0 \quad \text{for } x \leq \xi_1 \text{ and } x \geq \xi_K,$$

we arrive at:

For knots $\xi_1 < \xi_2 < \dots < \xi_K$, the K basis functions of natural cubic splines are:

$$N_1(X) = 1, \quad N_2(X) = X,$$

$$N_{k+2}(X) = d_k(X) - d_{K-1}(X), \quad k = 1, \dots, K - 2, \quad (3)$$

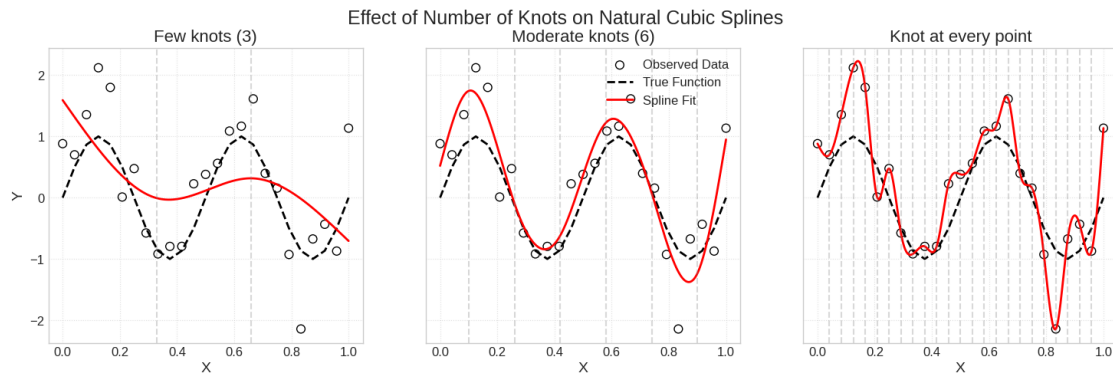
where

$$d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}. \quad (4)$$

2.4 Demo: Fitting with Natural Cubic Splines

Natural cubic splines often give superior fits at the boundaries, but how many knots and which should we pick?

- We generate noisy observations from the function $y = \sin(4\pi x)$.
- We fit **natural cubic splines** with different numbers of knots:
 - **Few knots (3):** The spline is very smooth and underfits the oscillations.
 - **Moderate knots (6):** The spline balances smoothness and flexibility, capturing the main trend.
 - **Knot at every data point:** The spline interpolates almost exactly, leading to overfitting.



2.5 B-Splines (Can Be Skipped)

Truncated power bases are numerically unstable for large numbers of knots. B-splines provide a basis with better numerical properties.

Definition 2.4 (B-Spline Basis). Each basis function $B_{i,m}(x)$ has local support over at most $m + 1$ knots.

Properties of B-Splines

- $B_{i,m}(x) \geq 0$ for all i, m, x .
- $B_{i,m}(x) = 0$ unless $\xi_i \leq x \leq \xi_{i+m+1}$ (local support).
- $\sum_i B_{i,m}(x) = 1$ for all x (partition of unity).
- They have the smallest support among all positive spline bases.

Computation: B-splines can be computed efficiently using the stable recurrence relations of de Boor (1978).

Advantage: Changing one coefficient affects the spline function only locally.

2.6 Smoothing Splines

Different approach: Instead of fixing the knots in advance, use the maximal set of knots and control the function's smoothness by regularization².

Definition 2.5 (Smoothing Spline). Among all functions f with two continuous derivatives, find the one that minimizes the penalized residual sum of squares:

$$\text{PRSS}(f, \lambda) = \sum_{i=1}^N (y_i - f(x_i))^2 + \lambda \int f''(t)^2 dt, \quad (5)$$

where $\lambda \geq 0$ is a fixed smoothing parameter.

- The first term measures closeness to the data.
- The second term is a **roughness penalty**. It measures the total curvature of the function.

²Analogously, we can consider penalized negative log-likelihood (for logistic regression, see ESL Ch. 5.6).

- The smoothing parameter controls the bias-variance trade-off.

2.7 Solution

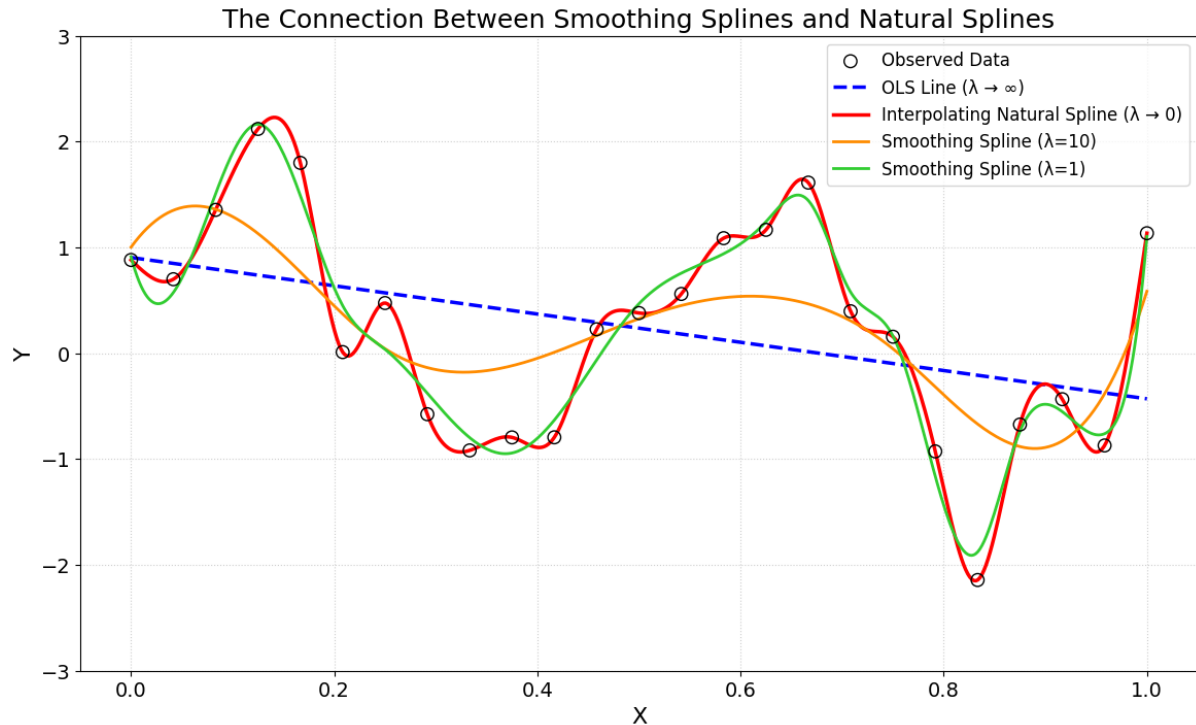
Theorem 2.6 (Solution to Smoothing Spline Problem). *The minimizer of $PRSS(f, \lambda)$ is a natural cubic spline with knots at the unique values of the $x_i, i = 1, 2, \dots, N$.*

Proof. See Exercise 1. □

Key points:

- $\lambda = 0$: interpolating spline (overfitting)
- $\lambda = \infty$: linear least-squares fit
- The solution is a finite-dimensional natural spline despite the infinite-dimensional optimization (over a Sobolev space of functions).

2.8 Illustration (Using the Same Data As Before)



2.9 Fitting Smoothing Spline in Basis Expansion

Let \mathbf{N} be the $N \times N$ matrix of natural spline basis functions evaluated at the training points x_i (spline analogue of regression design matrix).

Theorem 2.6 implies that we can write the fitted smoothing spline as

$$\hat{f}(x) = \sum_{j=1}^N N_j(x) \hat{\theta}_j, \quad (6)$$

where

$$\hat{\boldsymbol{\theta}} = (\mathbf{N}^T \mathbf{N} + \lambda \boldsymbol{\Omega}_N)^{-1} \mathbf{N}^T \mathbf{y} \quad (7)$$

and

$$(\boldsymbol{\Omega}_N)_{jk} = \int N_j''(t) N_k''(t) dt.$$

The fitted values are given by:

$$\hat{\mathbf{y}} = \mathbf{N}(\mathbf{N}^T \mathbf{N} + \lambda \boldsymbol{\Omega}_N)^{-1} \mathbf{N}^T \mathbf{y} = \mathbf{S}_\lambda \mathbf{y}, \quad (8)$$

where \mathbf{S}_λ is the **smoother matrix**. A smoothing spline with prechosen λ is an example of a linear smoother.

2.10 Properties and Effective Degrees of Freedom

Proposition 2.7 (Properties of Smoother Matrix). \mathbf{S}_λ is symmetric and positive semidefinite with eigenvalues in $[0, 1]$.

Proof. See blackboard. □

💡 To better see how smoothing controls the fit, we can write \mathbf{S}_λ in the Reinsch form

$$\mathbf{S}_\lambda = (\mathbf{I} + \lambda \mathbf{K})^{-1},$$

where \mathbf{K} does not depend on λ (see Exercise 2) and is known as the penalty matrix since $\hat{\mathbf{f}} = \mathbf{S}_\lambda \mathbf{y}$ solves

$$\min_{\mathbf{f}} (\mathbf{y} - \mathbf{f})^T (\mathbf{y} - \mathbf{f}) + \lambda \mathbf{f}^T \mathbf{K} \mathbf{f}.$$

Bias-Variance Tradeoff: The effective degrees of freedom

$$df_\lambda := \text{tr}(\mathbf{S}_\lambda)$$

measures the effective number of parameters in the smoothing spline fit.

- $\lambda = 0$: $df_\lambda = N$ (interpolating spline)
- $\lambda \rightarrow \infty$: $df_\lambda = 2$ (linear fit)

*Here, we focus on 1D spline models. For multidimensional analogs, see ESL Ch. 5.7.

3 Local Methods That Refit Everywhere: Local Averaging & Kernel Smoothing

3.1 Local Averaging Methods

Instead of fitting a single global function, we can compute a separate fit at each target point x_0 using only the training data that is **local** to that point.

Definition 3.1 (Local Averaging Estimators). A local averaging estimator has the general

form:

$$\hat{f}(x_0) = \sum_{i=1}^N w_i(x_0) y_i, \quad (9)$$

where the weights $w_i(x_0)$ depend on the target point x_0 .

- **Nearest Neighbors:** The weights are $w_i(x_0) = \frac{1}{k}$ if x_i is one of the k nearest neighbors of x_0 , and 0 otherwise.
- **Kernel Smoothing:** The weights are defined by a kernel function that smoothly down-weights points as their distance from x_0 increases, ensuring that the resulting estimated function is smooth across the input space.

💡 They are memory-based, non-parametric methods: they require little or no training; most of the computational work occurs at the time of evaluation (prediction). The entire training dataset acts as the model.

3.2 Smoothing Splines are Kernel Smoothers

- The smoother matrix \mathbf{S}_λ defines a set of weights for each target point. The prediction at a test point x_0 is a weighted average of all y_j .
- For any linear smoother \mathbf{S} , the prediction at a point x_0 can be written as a locally weighted average:

$$\hat{f}(x_0) = \sum_{i=1}^N L(x_0, x_i) y_i.$$

The function $L(x_0, \cdot)$ is called the **equivalent kernel** of the smoother.

- Denoting $N(x)$ as the column vector of basis functions evaluated at x , the smoothing spline can be written as

$$\hat{f}(x_0) = \sum_{i=1}^N L(x_0, x_i) y_i = L(x_0, \cdot)^T \mathbf{y},$$

with

$$L(x_0, \cdot)^T = N(x_0)^T (\mathbf{N}^T \mathbf{N} + \lambda \mathbf{\Omega}_N)^{-1} \mathbf{N}^T.$$

Equivalently, we can define

$$K(x, x') = N(x)^T (\mathbf{N}^T \mathbf{N} + \lambda \mathbf{\Omega}_N)^{-1} \mathbf{N}(x')$$

so that $L(x_0, x_i) = K(x_0, x_i)$. The equivalent kernel is just this bilinear form in the basis vectors.

- **The deep connection:** The smoothing spline, which we derived from a global, penalized regression problem, is in fact a linear kernel smoother³. Its equivalent kernel is *locally adaptive* (the bandwidth of the kernel is wider in regions where the data are more sparse).

3.3 A High Level Overview

In general, a penalized global basis expansion in the primal space can be equivalent to a local kernel smoother⁴ in the dual space.

³See: B. W. Silverman. "Spline Smoothing: The Equivalent Variable Kernel Method." *Ann. Statist.* 12 (3) 898–916, September, 1984.

⁴For nonlinear smoothers (wavelet smoothers, k -NNs, trees, neural nets), the mapping $\mathbf{y} \mapsto \hat{f}(x_0)$ is not linear in \mathbf{y} and the effective "weights" depend on the y_i , so a fixed equivalent kernel does not exist.

Aspect	Primal (linear basis model)	Dual (linear kernel smoother)
Unknowns	basis coefficients $\theta \in \mathbb{R}^M$	weights L on training outputs y_i
Function form	$\hat{f}(x) = \sum_{j=1}^M N_j(x) \theta_j$	$\hat{f}(x) = \sum_{i=1}^N L(x, x_i) y_i$
Penalty effect	Shrinks high-curvature coefficients	Determines locality/decay of $L(x, x_i)$
Computation	Solve $M \times M$ linear system	Solve $N \times N$ linear system
Intuition	Global smoothing	Local weighted averaging

💡 In the dual view, the basis expansion coefficients never need to be computed explicitly and everything is determined by inner products in a transformed feature space, where the transformation encodes the smoothing penalty. This idea is central to the theory of SVMs and kernel methods (next lecture).

3.4 One-Dimensional Kernel Smoothers

Let us look at a few important examples of kernel smoothers.

Definition 3.2 (The Nadaraya-Watson Kernel Estimator). The prediction at a point x_0 is a weighted average of the observed y_i :

$$\hat{f}(x_0) = \frac{\sum_{i=1}^N K_h(x_0, x_i) y_i}{\sum_{j=1}^N K_h(x_0, x_j)}, \quad (10)$$

where

$$K_h(x_0, x) = \frac{1}{h} K\left(\frac{|x - x_0|}{h}\right)$$

is a kernel function with bandwidth h .

Common Kernel Functions (typically symmetric, non-negative functions that integrate to one):

Epanechnikov:

$$K(t) = \frac{3}{4}(1 - t^2) \quad \text{if } |t| \leq 1, \quad 0 \text{ otherwise}$$

(see Exercise 3)

Tri-Cube:

$$K(t) = (1 - |t|^3)^3 \quad \text{if } |t| \leq 1, \quad 0 \text{ otherwise}$$

Gaussian:

$$K(t) = \phi(t)$$

where ϕ is the standard normal density

3.5 Local Polynomial Regression: Correcting Boundary Bias

The Nadaraya-Watson estimator, which fits a local constant, suffers from significant bias at the boundaries of the data. This is because its averaging window becomes one-sided.

Definition 3.3 (Local Polynomial Regression). Instead of a local constant, we fit a local

polynomial model at each target point x_0 by solving a weighted least-squares problem:

$$\min_{\beta(x_0)} \sum_{i=1}^N K_h(x_0, x_i) \left(y_i - \sum_{j=0}^d \beta_j(x_0) (x_i - x_0)^j \right)^2. \quad (11)$$

The most important case is local linear regression.

3.6 Local Linear Regression

Definition 3.4 (Local Linear Regression). At each target point x_0 , solve:

$$\min_{\alpha, \beta} \sum_{i=1}^N K_h(x_0, x_i) [y_i - \alpha(x_0) - \beta(x_0)x_i]^2. \quad (12)$$

The estimate is

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0.$$

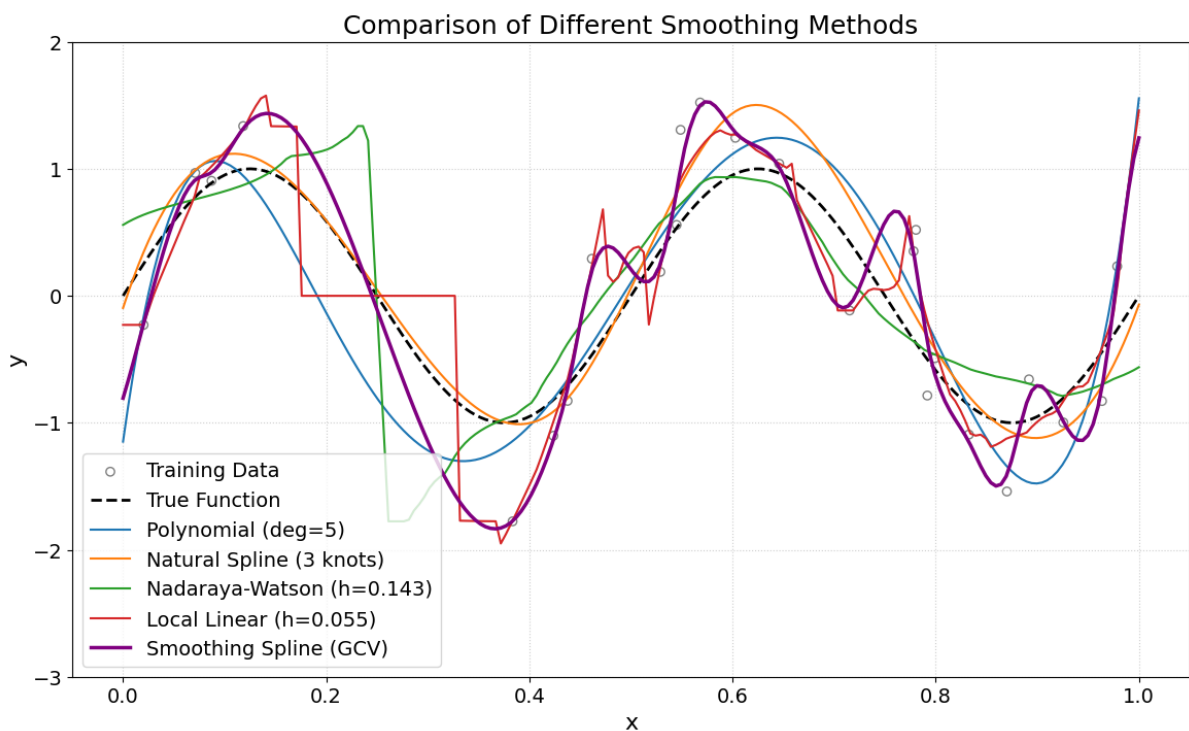
The local linear estimate can be written as

$$\hat{f}(x_0) = \mathbf{l}(x_0)^T \mathbf{y},$$

where $\mathbf{l}(x_0)$ are the equivalent kernel weights.

- Nadaraya-Watson has bias of order $O(h^2)$ in the interior (see Appendix) but $O(h)$ at boundaries (why?). Local linear regression has bias of order $O(h^2)$ everywhere, including boundaries.
- Local linear fits can help reduce bias dramatically at the boundaries at a modest cost in variance.

3.7 Comparison Demo (Using the Same Data As Before)



4 Kernel Density Estimation and Classification

4.1 Kernel Density Estimation: Use Kernels to Estimate PDFs

Definition 4.1 (Kernel Density Estimator (KDE) – Parzen Estimate). Given i.i.d. samples x_1, \dots, x_N from density $f(x)$,

$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^N K\left(\frac{x - x_i}{h}\right) =: \frac{1}{N} \sum_{i=1}^N K_h(x - x_i), \quad (13)$$

where K is a probability density function and $h > 0$ is the bandwidth controlling smoothness.

- $\hat{f}(x) \geq 0$ and $\int \hat{f}(x) dx = 1$.
- $\mathbb{E}[\hat{f}(x)] = (K_h * f)(x)$ (convolution with scaled kernel).
- As $h \rightarrow 0$ and $Nh \rightarrow \infty$, $\hat{f}(x) \rightarrow f(x)$ pointwise.
- **Connection to Nadaraya-Watson estimator:** See Exercise 4.
- **Naive Bayes classification via KDE:** For N_k points in class k , estimate class densities via

$$\hat{f}_k(x) = \frac{1}{N_k h} \sum_{x_i \in C_k} K\left(\frac{x - x_i}{h}\right).$$

Classify using

$$\hat{G}(x) = \arg \max_k \hat{\pi}_k \hat{f}_k(x),$$

where $\hat{\pi}_k = N_k/N$.

4.2 Radial Basis Functions (RBF)

Definition 4.2 (Radial Basis Function Expansion).

$$f(x) = \sum_{m=1}^M K_{h_m}(\mu_m, x) \beta_m, \quad (14)$$

where $K_{h_m}(\mu_m, \cdot)$ is a kernel function centered at μ_m with scale parameter h_m .

Gaussian RBF is a popular choice:

$$K_h(\mu, x) = e^{-\frac{\|x - \mu\|^2}{2h^2}},$$

with the resulting $f(x)$ representing functions as expansions in basis functions. The parameters are:

- Centers μ_m (often chosen as a subset of training points)
- Scales h_m (often kept constant)
- Coefficients β_m (estimated via least squares)

- **Extension to mixture models:** See ESL Ch. 6.8.

- **Key difference from kernel methods (later):** M is typically much smaller than N .

- **Connection to neural networks (later):** RBF networks are a type of neural network with one hidden layer, but with radial basis activation functions instead of sigmoids.

4.3 Practical Aspect: Selecting the Width of the Kernel

The bias-variance tradeoff:

- Small h : low bias, high variance (undersmoothing)
- Large h : high bias, low variance (oversmoothing)

Definition 4.3 (Leave-One-Out Cross-Validation (LOOCV) to Select h). Choose h to minimize:

$$CV(h) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{f}_{-i}(x_i))^2, \quad (15)$$

where \hat{f}_{-i} is the estimate with the i -th observation removed.

💡 Efficient CV computation for the Nadaraya-Watson estimator:

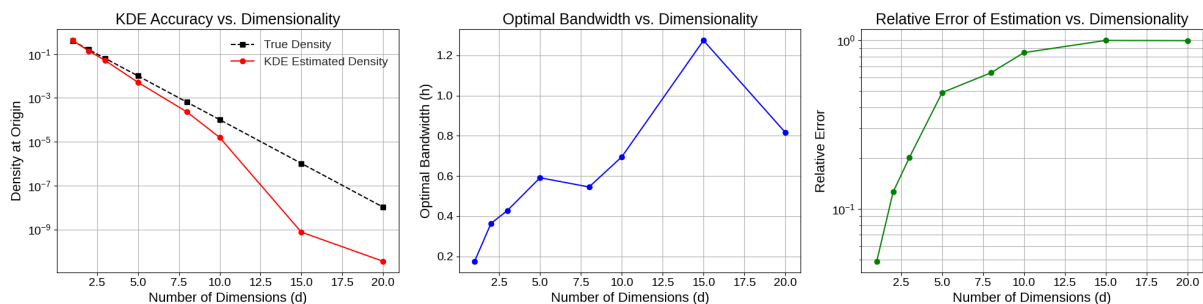
$$y_i - \hat{f}_{-i}(x_i) = \frac{y_i - \hat{f}(x_i)}{1 - S_{ii}(\lambda)},$$

where $S_{ii}(\lambda)$ is the i -th diagonal element of the smoother matrix. This allows CV computation with a single fit instead of N separate fits.

4.4 Curse of Dimensionality

Generate 2000 data points from the standard multivariate Gaussian and use KDE to estimate the PDF at the origin (mode of the distribution).

The Curse of Dimensionality for KDE (N = 2000 samples)



⚠️ **Over-estimation:** The KDE estimator⁵ is forced to select a huge bandwidth in high dimensions, reporting a smeared-out average over a large, non-local region.

4.5 Kernel Density Classification

Recall: The goal of classification is to estimate the probability that an observation X belongs to class k :

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}. \quad (16)$$

- $P(Y = k|X = x)$: The posterior probability — what we want.
- $\pi_k = P(Y = k)$: The prior probability of class k .
- $f_k(x) = p(x|Y = k)$: The class-conditional probability density.

⁵See <https://www.stat.cmu.edu/~larry/=sml/densityestimation.pdf> for a deeper dive into the fascinating topic of density estimation.

The Optimal Decision Rule: Assign x to the class k that maximizes the numerator:

$$\text{Class}(x) = \underset{k}{\operatorname{argmax}} (\pi_k f_k(x)).$$

4.6 The Core Challenge: Estimating the Density $f_k(x)$

The Bayes rule is perfect, but in practice, we *never know* the true class-conditional densities $f_k(x)$. We must estimate them from the training data.

1. Parametric Methods

- Assume a specific functional form for $f_k(x)$ (e.g., a multivariate Gaussian).
- Example: Linear Discriminant Analysis (LDA).
- **Pro:** Efficient if assumptions are correct.
- **Con:** High bias; performs poorly if assumptions are violated.

2. Non-Parametric Methods

- Do not assume a specific form for $f_k(x)$. Let the data speak for itself.
- Example: **Kernel Density Estimation (KDE)**.
- **Pro:** Highly flexible; can capture complex density shapes.
- **Con:** Requires more data; can be computationally intensive, and may be prone to the curse of dimensionality.

4.7 The General Kernel Density Classifier

This is the most direct non-parametric approach.

1. **Split Data:** Divide the training data by class (C_1, C_2, \dots, C_K).
2. **Estimate Priors:** Estimate $\hat{\pi}_k$ as the proportion of samples in class k , i.e. $\hat{\pi}_k = N_k/N$.
3. **Estimate Densities:** For each class k , use KDE on the data in C_k to get an estimate $\hat{f}_k(x)$.
4. **Classify:** For a new point x_{new} , apply Bayes' rule using these estimates:

$$\text{Class}(x_{\text{new}}) = \underset{k}{\operatorname{argmax}} \left(\hat{\pi}_k \hat{f}_k(x_{\text{new}}) \right).$$

⚠ This works well in low dimensions (1 or 2). But estimating the *joint* multivariate density $\hat{f}_k(x)$ with KDE suffers severely from the **curse of dimensionality**.

4.8 The Solution: The Naive Bayes Assumption

To overcome the curse of dimensionality, assume that within each class, the features are independent of one another (conditional independence). This allows us to factorize the joint class-conditional density $f_k(x)$ into a product of one-dimensional marginal densities:

$$f_k(x) = f_k(x_1, x_2, \dots, x_p) = \prod_{j=1}^p f_{kj}(x_j), \quad (17)$$

where $f_{kj}(x_j)$ is the density of the j -th feature for an observation from class k .

Why is this “Naive”?

- In most real-world problems, features are not truly independent (e.g., height and weight are correlated).
- However, the assumption often works remarkably well in practice and makes the problem tractable.

4.9 The Naive Bayes Classifier with KDE

Combines the flexibility of KDE with the dimensionality-reducing power of the Naive Bayes assumption.

1. **Estimate Priors:** As before, $\hat{\pi}_k = N_k/N$.
2. **Estimate Marginal Densities:** For each class k and for each feature j , use the data from class k and feature j to build a **1-dimensional KDE**, giving $\hat{f}_{kj}(x_j)$.
3. **Combine Densities:** The full class-conditional density estimate is:

$$\hat{f}_k(x) = \prod_{j=1}^p \hat{f}_{kj}(x_j).$$

4. **Classify:** Apply the Bayes rule with this factorized form:

$$\text{Class}(x_{\text{new}}) = \underset{k}{\operatorname{argmax}} \left(\hat{\pi}_k \prod_{j=1}^p \hat{f}_{kj}(x_{\text{new},j}) \right).$$

**In practice, we compute the sum of log-probabilities to avoid numerical underflow.*

4.10 Effect of the Independence Assumption in Naive Bayes

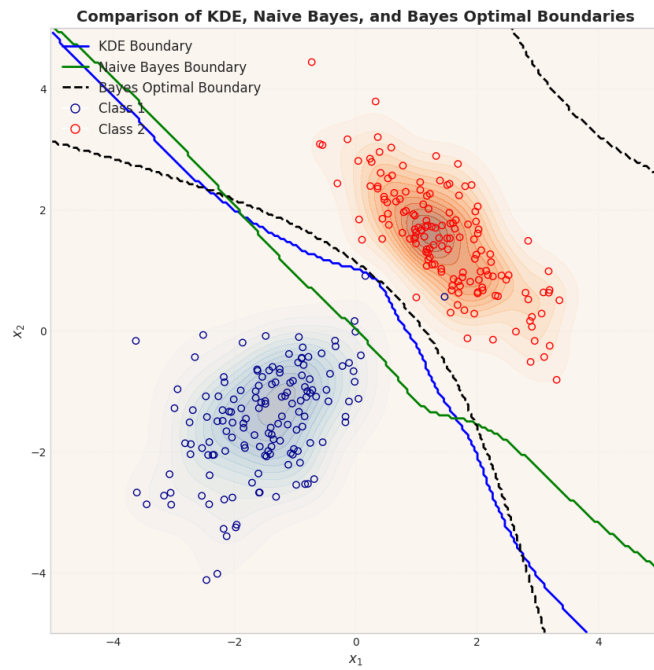
Two classes of 2D data, sample 150 samples from each for training.

Class 1:

$$X \sim \mathcal{N} \left([-1.5, -1.5], \begin{bmatrix} 0.8 & 0.4 \\ 0.4 & 0.8 \end{bmatrix} \right),$$

Class 2:

$$X \sim \mathcal{N} \left([1.5, 1.5], \begin{bmatrix} 0.8 & -0.6 \\ -0.6 & 0.8 \end{bmatrix} \right).$$



💡 For classification, accurately estimating posterior probabilities near the decision boundary is most critical.

5 Exercises

1. Solve Exercise 5.7 in ESL.
2. Solve Exercises 5.9 and 5.11 in ESL. Discuss the implications of these results.
3. (a) Verify that the Epanechnikov kernel function is a symmetric probability distribution function.
(b) Solve Exercise 6.1 in ESL.
4. Solve Exercise 6.8 in ESL.
5. **Analysis of Linear Kernel Smoothers in the Fixed Design Setting.** Solve Exercise 6.10 in ESL.

[Experimental] Generate a simple one-dimensional dataset with $N = 50$ points from the model:

$$y = \sin(2\pi x^3) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 0.2^2),$$

where x is drawn uniformly from $[0, 1]$.

- Fit a global polynomial of degree 5.
- Fit a natural cubic spline with knots placed at the 25th, 50th, and 75th percentiles of the data.
- Fit a Nadaraya-Watson (local constant) kernel smoother.
- Fit a local linear kernel smoother.
- Fit a smoothing spline.

For the above methods, use 5-fold CV to select the optimal bandwidth or effective degrees of freedom. Plot all five fitted curves on top of the training data and the true function. Discuss the results.

A Appendix

A.1 Appendix: Bias and Variance of Kernel Smoothers

For a deeper dive on kernel smoothers, see <https://www.stat.cmu.edu/~larry/=sml/nonpar2019.pdf>.

Assume $y = f(x) + \varepsilon$ where $\mathbb{E}[\varepsilon] = 0$ and $\text{Var}(\varepsilon) = \sigma^2$. For a test point x_0 ,

$$\text{MSE}(\hat{f}(x_0)) = (\mathbb{E}[\hat{f}(x_0)] - f(x_0))^2 + \text{Var}(\hat{f}(x_0)) = \text{squared bias} + \text{variance}. \quad (18)$$

Let \hat{f} be the Nadaraya-Watson kernel estimator with bandwidth h . Assume:

- The training points x_i are a fixed design with density $p(x)$.
- The true function $f(x)$ is twice continuously differentiable.
- The kernel $K(u)$ is a symmetric probability density with $\int uK(u)du = 0$.
- As $N \rightarrow \infty$, we have $h \rightarrow 0$ and $Nh \rightarrow \infty$.

Then, the leading terms for the bias and variance at x_0 are:

•

$$\text{Bias}^2(\hat{f}(x_0)) \approx \left[\frac{h^2}{2} \left(f''(x_0) + \frac{2f'(x_0)p'(x_0)}{p(x_0)} \right) \int u^2 K(u) du \right]^2 = O(h^4).$$

•

$$\text{Var}(\hat{f}(x_0)) \approx \frac{\sigma^2}{Nhp(x_0)} \int K(u)^2 du = O\left(\frac{1}{Nh}\right).$$

💡 Minimizing $h^4 \cdot C_1 + \frac{1}{Nh} \cdot C_2$ with respect to h shows that the **optimal bandwidth is of order** $N^{-1/5}$. This yields optimal error rate of $O(N^{-4/5})$.

A.2 Appendix: Derivation of Asymptotic Bias and Variance

The derivation requires approximating discrete sums with integrals. This is only valid under the assumption that as $N \rightarrow \infty$, we have $h \rightarrow 0$ and $Nh \rightarrow \infty$.

The expected value is

$$\mathbb{E}[\hat{f}(x_0)] = \frac{\sum_{i=1}^N K_h(x_0, x_i) f(x_i)}{\sum_{j=1}^N K_h(x_0, x_j)}.$$

Step 1: Taylor Expansion. We expand $f(x_i)$ around x_0 . This is only accurate if $|x_i - x_0|$ is small. Since the kernel's support is of width h , we require $\mathbf{h} \rightarrow \mathbf{0}$.

$$f(x_i) \approx f(x_0) + (x_i - x_0)f'(x_0) + \frac{(x_i - x_0)^2}{2}f''(x_0).$$

Step 2: Approximate Sums with Integrals. This step is valid because the number of points in the kernel's window is large, which requires $Nh \rightarrow \infty$. Let $u = (x - x_0)/h$.

$$\text{Denominator} = \sum_j K_h(x_j, x_0) \approx N \int K\left(\frac{x - x_0}{h}\right) p(x) dx = Nh \int K(u) p(x_0 + hu) du \approx Nhp(x_0).$$

Step 3: Combine and Solve. Applying the same steps to the numerator and using the symmetry of the kernel (so terms with odd powers of u vanish), the leading bias term is:

$$\text{Bias}(\hat{f}(x_0)) \approx \frac{h^2}{2} f''(x_0) \int u^2 K(u) du.$$

The variance is

$$\text{Var}(\hat{f}(x_0)) = \sigma^2 \frac{\sum_i K_h(x_0, x_i)^2}{(\sum_j K_h(x_0, x_j))^2}.$$

Again, we approximate the sums with integrals, which requires $Nh \rightarrow \infty$:

$$\begin{aligned} \text{Numerator Sum} &\approx N \int K\left(\frac{x-x_0}{h}\right)^2 p(x) dx \approx Nhp(x_0) \int K(u)^2 du, \\ \text{Denominator Sum}^2 &\approx (Nhp(x_0))^2. \end{aligned}$$

Combining these gives the final result:

$$\text{Var}(\hat{f}(x_0)) \approx \sigma^2 \frac{Nhp(x_0) \int K(u)^2 du}{(Nhp(x_0))^2} = \frac{\sigma^2}{Nhp(x_0)} \int K(u)^2 du.$$