STA 35C: Statistical Data Science III

Lecture 22: Smoothing Splines & Principal Component Analysis

Dogyoon Song

Spring 2025, UC Davis

Today's topics

- **Review**: Regression splines
  - Formulation & basic properties
  - Truncated power basis representation
  - "Natural" splines
  - How to place knots
- **Smoothing splines**
  - Overview: interpolation + smoothness penalty

    $\rightarrow$ The resulting function is a natural cubic spline!
  - Choosing the smoothness parameter
- **Unsupervised learning & Principal component analysis**
  - Overview of unsupervised learning
  - Principal components as directions capturing max variance

Quick review: Regression splines

Regression splines:

- *Piecewise polynomials* of degree d, joined smoothly at knots (cutpoints)
  - *Continuity constraints* at each knot for the function and its first $(d-1)$ derivatives
  - A degree-d spline with K knots has $(d+1)+K$ parameters
- *Truncated power basis* for degree d with knots at $c_1, \ldots, c_K$:

$$f(x) = \beta_0 + \beta_1 X + \cdots + \beta_d X^d \; + \; \sum_{k=1}^{K} \beta_{d+k} \, (X - c_k)_+^d$$

  - These basis functions automatically ensure continuity up to $(d-1)$-th derivative
- *Natural splines* add constraints so the spline is linear beyond the outer knots
  - For a cubic spline ($d=3$), this reduces DoF from $(K+4)$ to $(K+2)$
- *Placing knots*
  - Common schemes: uniform width vs. uniform mass (percentiles)
  - Often use cross-validation for final selection

Smoothing splines: Formulation

Goal: Estimate a function $g(x)$ that fits observed data (x_i, y_i) well, avoiding overfitting

Idea: Minimize a composite objective comprising data fidelity & smoothness penalty:

$$\min_{g \in \mathcal{G}} \left\{ \underbrace{\sum_{i=1}^{n} \left(y_i - g(x_i)\right)^2}_{\text{RSS; data fidelity}} + \lambda \underbrace{\int \left(g''(t)\right)^2 dt}_{\text{smoothness penalty}} \right\}$$

- The parameter $\lambda \geq 0$ balances data fit vs. smoothness
  - $\lambda = 0$: interpolates all points (often wiggly)
  - $\lambda \to \infty$: slope is constant, i.e., a straight least squares line

Result: The solution is a *natural cubic spline* with knots at each x_i

- However, somewhat *shrunken* relative to a plain regression spline

In R: The function `smooth.spline()` (in base R) performs smoothing spline fitting

Smoothing splines: Curvature penalty

Why penalize $\int (g''(t))^2 \, dt$?

- The penalty $\int (g'')^2$ aims to control "roughness" or high-frequency wiggles
  - The second derivative measures how sharply g bends
  - A more wiggly g has bigger $(g'')^2$, thus a larger penalty
- Minimizing $\int (g'')^2$ forces smaller curvature, yielding a smoother shape

(`Optional`) Food for thought*: What if we instead used $\int (g)^2$ or $\int (g')^2$?

- $\int (g)^2$ would penalize the "magnitude" of g
- $\int (g')^2$ would penalize "slopes"
- cf. Homework 5, Problem 3-3*

Regression splines vs. smoothing splines:

- **Regression splines**: fix knots/degree and enforce derivative continuity
- **Smoothing splines**: solve a *penalized* least squares problem; knots effectively spread out adaptively to balance data fit vs. smoothness.

Choosing the smoothing parameter λ

Effective degrees of freedom:

- As λ increases from 0 to ∞, the resulting regression function shifts from an *interpolation* spline (exactly fitting all data) to a simple *straight line*
- The *effective* degrees of freedom, df_λ, decreases from *n* to 2
  - Degrees of freedom $=$ the number of free parameters
  - The *n* parameters of smoothing spline are often "shrunk" due to penalty
- $\mathrm{df}_\lambda \in [2, n]$ quantifies the spline's complexity

Selecting λ (or df_λ):

- Typically use cross-validation; LOOCV is practical thanks to a short-cut formula
- In practice:
  - Pick a small grid of λ-values (or df_λ-values)
  - Compute CV error and select the minimizer
- **In R:** We specify df_λ, and **R** will compute corresponding λ

If interested, see [JWHT21, Sec. 7.5.2] for more technical details

Smoothing splines: Illustration

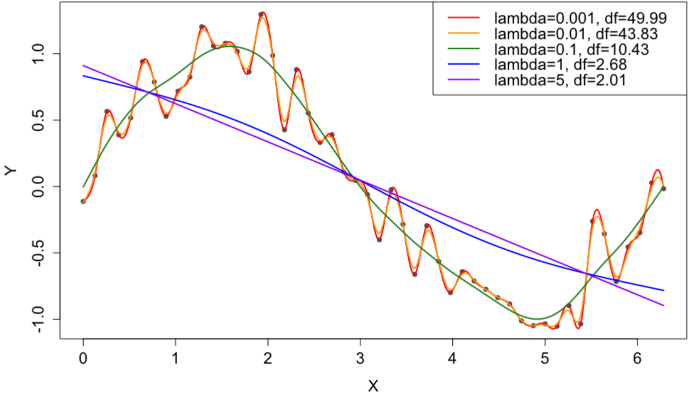

Figure: Smoothing splines fit to $n = 50$ data points from $Y = \sin(X) + \epsilon$ on $[0, 2\pi]$ at various λ values. For smaller λ, the fit nearly interpolates all points (wiggly). As λ grows larger, the fit becomes smoother, and for very large λ, it approximates a straight line.

Smoothing splines: A toy example code in R

```r
set.seed(123)
n <- 50
x <- seq(0, 2*pi, length.out = n)
y <- sin(x) + rnorm(n, sd = 0.2)

lambda_values <- c(0.001, 0.01, 0.1, 1, 5)

# smooth.spline doc says:
# lambda = 256^(spar - 1)
# => spar = 1 + log(lambda, base=256)
lambda_to_spar <- function(lambda) {
  1 + log(lambda, base = 256)
}

fits <- lapply(lambda_values, function(lam) {
  spar_val <- lambda_to_spar(lam)
  smooth.spline(x, y, spar = spar_val)
})
```

For additional examples, see the discussion section materials (Week 8) and [JWHT21, Sec. 7.8.1 & 7.8.2]

```r
plot(x, y,
     pch = 16, col = "gray40",
     main = "",
     xlab = "X", ylab = "Y",
     cex.axis = 1.4,
     cex.lab = 1.4)

xx <- seq(min(x), max(x),
          length.out = 400)
colors <- c("red","orange",
           "forestgreen",
           "blue","purple")

for(i in seq_along(lambda_values)) {
  sfit <- fits[[i]]
  yy <- predict(sfit,
               x=xx)$y
  lines(xx, yy,
        col=colors[i],
        lwd=2)
}
```

Pop-up quiz #2: Smoothing splines

Which statement is false about smoothing splines?

A) They solve a penalized least squares problem with a curvature penalty, $\int (g''(t))^2 dt$.

B) As the smoothing parameter $\lambda \to 0$, the spline interpolates all data points, possibly becoming wiggly.

C) As $\lambda \to \infty$, the spline degenerates to a simple linear fit.

D) The effective degrees of freedom always remains fixed at 4 for any smoothing spline fit.

Answer: (D) is false.

The effective degrees of freedom for a smoothing spline varies between n (very wiggly, when $\lambda = 0$) and 2 (almost a straight line, as $\lambda \to \infty$), not a fixed value of 4.

Regression & smoothing splines: Summary

- **Regression splines:**
  - Piecewise polynomials + continuity constraints
  - Truncated power basis for an easy linear-model fit
  - "Natural" splines impose linear boundary conditions to avoid erratic tails
  - Choose knot placement / number of knots via cross-validation
- **Smoothing splines:**
  - A penalized approach balancing data fit vs. curvature
  - The solution is a natural cubic spline with knots at each data point
  - $\lambda \to 0$ yields interpolation; $\lambda \to \infty$ yields a line
  - Effective degrees of freedom: from n to 2
  - Typically choose λ by cross-validation

Unsupervised learning

Contrast with supervised learning:

- Supervised learning: we observe (X, Y), and want to learn a function $f : X \to Y$
- *Unsupervised learning*: only predictors $X = (X_1, \ldots, X_p)$; *no* response data

Goal of unsupervised learning:

- Perform *exploratory* analysis of X to discover patterns or structures in the data
- Examples:
  - Cancer research: Identify subgroups among patients/genes using gene-expression profiles
  - Online shopping: Group shoppers with similar purchase patterns, then recommend items
  - Search engines: Tailor search results to individuals with similar click histories

We will study two types of unsupervised learning tasks/methods:

- *Principal component analysis (PCA):* Find a few directions capturing most variation in the data
- *Clustering:* Identify subgroups (clusters) among observations

Principal component analysis (PCA): Overview

Goal: Reduce the dimension of $X = (X_1, \ldots, X_p)$ from p to $r \ll p$ while preserving as much variability in data as possible

Reasons for dimensionality reduction:

- *Statistical efficiency:* Avoid overfitting and high variance in high-dimensional spaces
- *Computational benefits:* Easier/faster to store and process fewer variables
- *Visualization:* Plotting or interpreting data in 2D or 3D
- *Noise reduction:* Focusing on major signals in the data

How does PCA work?

- Identifies a few orthogonal directions that capture the maximum variance in the data
- These directions are called the *principal components (PCs)*

PCA: Visual Illustration for $p = 2$

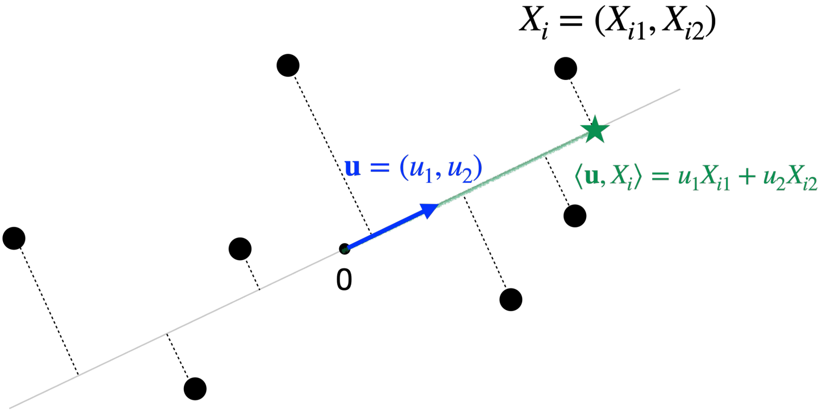

Figure: For any unit vector $\mathbf{u} = (u_1, u_2)$, consider the orthogonal projection of X_i onto the line spanned by $\mathbf{u}$, which is $\langle \mathbf{u}, X_i \rangle = u_1 X_{i1} + u_2 X_{i2}$. PCA finds the direction $\mathbf{u}$ that maximizes the variance of $\langle \mathbf{u}, X_i \rangle$.

PCA: Formulation ($p = 2$)

Assumption: Data is *centered*, i.e. $\mathbb{E}[X] = \frac{1}{n}\sum_{i=1}^{n} X_i = 0$

Key idea of PCA:

- Pick a *direction* $\mathbf{u} = (u_1, u_2)$ with $\|\mathbf{u}\|^2 = u_1^2 + u_2^2 = 1$
- The *variance* of X along $\mathbf{u}$ is

$$\mathrm{Var}(\langle \mathbf{u}, X\rangle) = \mathbb{E}\left[(\langle \mathbf{u}, \mathbf{X}\rangle - \mathbb{E}\langle \mathbf{u}, X\rangle)^2\right] = \frac{1}{n}\sum_{i=1}^{n}(u_1 x_{i1} + u_2 x_{i2})^2.$$

- **1st principal component** $=$ the direction $\mathbf{u}$ that *maximizes* this variance:

$$\text{maximize } \frac{1}{n}\sum_{i=1}^{n}(\mathbf{u}\cdot\mathbf{x}_i)^2 \quad \text{subject to} \quad \|\mathbf{u}\| = 1$$

- Geometrically, this is like finding the "major axis" of an ellipsoid formed by the data

PCA: General formulation

For higher dimensions ($p \geq 2$):

- **First principal component:** a unit vector $\mathbf{u}_1 \in \mathbb{R}^p$ that solves

$$\text{maximize } \frac{1}{n} \sum_{i=1}^{n} (\mathbf{u}_1 \cdot \mathbf{x}_i)^2 \quad \text{subject to} \quad \|\mathbf{u}_1\| = 1$$

- **Second principal component:** a unit vector $\mathbf{u}_2 \in \mathbb{R}^p$ orthogonal to $\mathbf{u}_1$ that maximizes the projected variance, and so on
- (`Optional`) Equivalently, the principal components are the eigenvectors of the sample covariance matrix of X
  - The eigenvalues indicate the variance each principal component captures

Key points:

- PCA is unsupervised (no Y)
- It finds linear combinations of predictors (features) capturing maximum variance
- The first few principal components ($r \ll p$) often capture most of the total variation, enabling dimension reduction

References

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani.
An Introduction to Statistical Learning: with Applications in R, volume 112 of *Springer Texts in Statistics*.
Springer, New York, NY, 2nd edition, 2021.