A Crash Course on P-splines

Paul Eilers & Brian Marx

Channel Network Conference Nijmegen
April 2015
A Crash Course on P-splines

Introduction
What are P-splines?

- A flexible tool for smoothing
- Based on regression
- Local basis functions: B-splines
- No efforts to optimize the basis
- Just a large bunch of B-splines
- And a penalty to tune smoothness
- (Software demo: PSPlay_psplines)

The roots of P-splines

- Eilers and Marx: Statistical Science, 1996
- In fact not a very revolutionary proposal
- A simplification of O’Sullivan’s ideas
- But the time seemed right
- Now almost 1000 citations
- Many from applied areas (what really counts for us)
- E&M evangelized heavily
- A variety of applications, many in this course

The plan of the course

- We start with penalties
- They are the core ingredient
- Splines come later
- They just “add the flesh to the skeleton”
- Basic (generalized) linear smoothing
- Extensions: generalized additive models, 2-D smoothing
- Bayesian and mixed model interpretations
- Specialized penalties
Part 1

The power of penalties
Discrete smoothing

- Given: data series \(y_i, i = 1, \ldots, m\)
- Wanted: a smooth series \(z\)
- Two (conflicting) goals: fidelity to \(y\) and smoothness
- Fidelity, sum of squares: \(S = \sum_i (y_i - z_i)^2\)
- How to quantify smoothness?
- Use roughness instead: \(R = \sum_i (z_i - z_{i-1})^2\)
- Simplification of Whittaker’s (1923) “graduation”

Penalized least squares

- Combine fidelity and roughness
  \[
  Q = S + \lambda R = \sum_i (y_i - z_i)^2 + \lambda \sum_i (z_i - z_{i-1})^2
  \]
- Parameter \(\lambda\) sets the balance
- Operator notation: \(\Delta z_i = z_i - z_{i-1}\)
  \[
  Q = \sum_i (y_i - z_i)^2 + \lambda \sum_i (\Delta z_i)^2
  \]

Matrix-vector notation

- Penalized least squares objective function
  \[
  Q = ||y - z||^2 + \lambda ||Dz||^2
  \]
- Differencing matrix \(D\), such that \(Dz = \Delta z\)
  \[
  D = \begin{bmatrix}
  -1 & 1 & 0 & 0 \\
  0 & -1 & 1 & 0 \\
  0 & 0 & -1 & 1 \\
  \end{bmatrix}
  \]
- Explicit solution: \(\hat{z} = (I + \lambda D'D)^{-1} y\)

Implementation in R

```r
m <- length(y)
E <- diag(m)  # Identity matrix
D <- diff(E)  # Difference operator
G <- E + lambda * t(D) %*% D
z <- solve(G, y)  # Solve the equations
```
Notes on computation

- Linear system of equations
- \(m\) equations in \(m\) unknowns
- Practical limit with standard algorithm: \(m \approx 4000\)
- System is extremely sparse (bandwidth = 3)
- Specialized algorithms easily handle \(m > 10^6\)
- Computation time then linear in \(m\)

Higher order penalties

- Higher order differences are easily defined
- Second order: \(\Delta^2 z_i = \Delta(\Delta z_i) = (z_i - z_{i-1}) - (z_{i-1} - z_{i-2})\)
- Second order differencing matrix
  \[
  D = \begin{bmatrix}
  1 & -2 & 1 & 0 & 0 \\
  0 & 1 & -2 & 1 & 0 \\
  0 & 0 & 1 & -2 & 1
  \end{bmatrix}
  \]
- Higher orders are straightforward
- In R: \(D = \text{diff(diag(m), diff = d)}\)

The effects of higher orders

- Smoother curves
- Polynomial limits for large \(\lambda\)
- Degree of interpolation
- Degree of extrapolation
- Conservation of moments (will be explained later)
- (Software demo: PSPlay_discrete)
Limits

- Consider large $\lambda$ in $Q = \|y - z\|^2 + \lambda\|Dz\|^2$
- Penalty is overwhelming, hence essentially $Dz = \Delta z = 0$
- This is the case if $z_i - z_{i-1} = 0$, hence $z_i = c$, a constant
- Generally: $\Delta^d z = 0$ if $z$ is order $d - 1$ polynomial in $i$
- Linear limit when $d = 2$, quadratic when $d = 3$, ...
- It is also the least squares polynomial

Interpolation and extrapolation

- Let $y_i$ be missing for some $i$
- Use weights $w_i$ (0 if missing, 1 if not)
- Fill in arbitrary values (say 0) for missing $y$
- Minimize, with $W = \text{diag}(w)$

$$Q = (y - z)'W(y - z) + \lambda\|Dz\|^2$$

- Trivial changes: $\hat{z} = (W + \lambda D'D)^{-1}Wy$

Interpolation and extrapolation, continued

- Interpolation is by polynomial in $i$
- Order $2d - 1$
- Extrapolation: introduce “missing” data at the end(s)
- Extrapolation is by polynomial in $i$
- Order $d - 1$
- (Software demo: PSPlay_interpolate program)

Plot from PSPlay interpolate program

Whittaker smoothing; order = 2, log10(lambda) = 2.4
Non-normal data

- We measured fidelity by the sum of squares
- This is reasonable for (approximately) normal data
- Which means: trend plus normal disturbances
- How will we handle counts?
- Or binomial data?
- Use penalized (log-)likelihood
- Along the lines of the generalized linear model (GLM)

Smoothing of counts

- Given: a series \( y \) of counts
- We model a smooth linear predictor \( \eta \)
- Assumption: \( y_i \sim \text{Pois} (\mu_i) \), with \( \eta_i = \log \mu_i \)
- The roughness penalty is the same
- But fidelity measured by deviance (-2 LL):
  \[
  Q = 2 \sum_i (\mu_i - y_i \eta_i) + \lambda \sum_i (\Delta^d \eta_i)^2
  \]

Linearization and weighted least squares

- Derivatives of \( Q \) give penalized likelihood equations
  \[
  \lambda D' D z = y - e^\eta = y - \mu
  \]
- Non-linear, but the Taylor approximation gives
  \[
  (\tilde{M} + \lambda D' D) \eta = y - \tilde{\mu} + \tilde{M} \tilde{\eta}
  \]
- Current approximation \( \tilde{\eta} \), and \( \tilde{M} = \text{diag}(\tilde{\mu}) \)
- Repeat until (quick) convergence
- Start from \( \tilde{\eta} = \log(y + 1) \)

Example: coal mining accidents

1860 1880 1900 1920 1940 1960
0 1 2 3 4 5 6
Year
# of disasters

Mining disasters; order = 2, \( \log_{10}(\lambda) = 1 \)
A useful application: histogram smoothing

- The “Poisson smoother” is ideal for histograms
- Bins can be very narrow
- Still a smooth realistic (discretized) density estimate
- Conservation of moments
- \( \sum_i y_i x_i^k = \sum_i \hat{\mu}_i x_i^k \) for integer \( k < d \) (bin midpoints in \( x \))
- With \( d = 3 \), mean and variance don’t change
- Whatever the amount of smoothing
- (Software demo: PSPlay_histogram)

Plot from PSPlay_histogram program

Respect the boundaries
- Extend the histogram with enough zero counts
- But some data are inherently bounded
- Non-zero, or between 0 and 1
- Then you should limit the domain accordingly
- Otherwise you will smooth in the “no go” area
- Example: suicide treatment data
- Inherently non-negative durations

Smoothing old Faithful

Old Faithful; order = 2, log10(lambda) = 1

Histogram smoothing; order = 2, log10(lambda) = 3

Frequency

-3 -2 -1 0 1 2 3

0 2 4 6 8

Eruption length (min)
### Smoothing the suicide treatment data

- **Graph:**
  - **Title:** Suicide treatments, order = 2, log10(lambda) = 1
  - **X-axis:** Treatment spells (d)
  - **Y-axis:** Frequency
  - **Data Points:**
    - Frequency 0: 200, 400, 600, 800, 1000
    - Frequency 5: 0, 5, 10, 15

### Binomial data

- **Given:** sample sizes $s$, “successes” $y$
- **Smooth curve wanted for $p$, probability of success**
- **We model the logit:**
  
  \[ z = \log \frac{p}{1 - p}; \quad p = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}} \]

- **Linearization as for counts**
- **Start from logit of $(y + 1)/(s + 2)$**
- **No surprises, details skipped**

### Example: hepatitis B prevalence (Keiding)

- **Graph:**
  - **Title:** Hepatitis B prevalence
  - **X-axis:** Age (yr)
  - **Y-axis:** Prevalence
  - **Data Points:**
    - Age 0: 20, 40, 60, 80
    - Prevalence 0.0: 0.0, 0.2, 0.4, 0.6, 0.8, 1.0

### Optimal smoothing

- **We can smooth almost anything (in GLM sense)**
- **How much should we smooth?**
- **Let the data decide**
- **Cross-validation, AIC (BIC)**
- **Essentially we measure prediction performance**
- **On new or left-out data**
Leave-one-out cross-validation

• Leave out $y_i$ (make $w_i$ zero)
• Interpolate a value for it: $\hat{y}_{-i}$
• Do this for all observations in turn
• You get a series of “predictions”
• How good are they?
• Use $CV = \sum (y_i - \hat{y}_{-i})^2$, or $RMSCV = \sqrt{CV/m}$
• Search for $\lambda$ that minimizes $CV$

Speeding up the computations

• LOO CV looks expensive (repeat smoothing $m$ times)
• It is, if done without care
• But there is a better way
• We have $\hat{y} = (W + \lambda D' D)^{-1} W y = H y$
• We call $H$ the hat matrix; property: $h_{ij} = \partial \hat{y}_i / \partial y_j$
• One can prove: $y_i - \hat{y}_{-i} = (y_i - \hat{y}_i) / (1 - h_{ii})$
• Smooth once (for each $\lambda$), compute all $\hat{y}_{-i}$ at the same time

Akaike’s information criterion

• Definition: $AIC = \text{Deviance} + 2ED = -2LL + 2ED$
• Here $ED$ is the effective model dimension
• Useful definition:
  $$ED = \sum_i \frac{\partial \hat{\mu}_i}{\partial y_i} = \sum_i h_{ii} = \text{tr}(H)$$
• This defines a hat matrix for generalized linear smoothing
• Vary $\lambda$ on a grid to find minimum of AIC
• Minimization routine can be used too
• But it is useful to see the curve of AIC vs. $\log \lambda$

A convincing example: Old Faithful

Old Faithful; order = 2, log10($\lambda$) = 1

AIC

Eruption length (min)
Frequency

Old Faithful; order = 2, log10($\lambda$) = 1

AIC vs. log10($\lambda$)
What went wrong?

- The (silent) assumption: trend plus independent noise
- Here the noise is correlated
- LOO CV means: best prediction of left-out data
- Light smoothing gives better predictions
- That is not what we had in mind
- The smooth trend is not automatically detected

Two solutions

- The elegant solution: model correlated noise
- This has been done (Currie and Durban)
- A lot of extra work
- Simple alternative: take every fifth (tenth) observation
- Thinning observations breaks correlation
- Scale final $\lambda$ by $f^2d$
- If $f$ is the thinning factor

Thinning to break correlation
Similar problems with histograms

- If counts are a time series, AIC can fail
- Again serial correlation is the cause
- Other histograms show digit preference
- People read an analog scale or estimate a number
- Examples: blood pressure in mm (mostly even numbers)
- Age, or birth date: rounding to multiples of five.
- Solution: model digit preference (non-trivial)
- Or use your carpenter’s eye

Circular smoothing

- Sometimes the data are circular
- Because we look at one period (or more)
- Then we wish that both ends connect smoothly
- Modify difference matrix with extra row(s), like

\[
D = \begin{bmatrix}
-1 & 1 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & -1 & 1 \\
1 & 0 & 0 & 0 & -1
\end{bmatrix}
\]

Illustration of circular smoothing

Designer penalties

- By now you should have got the message
- The penalty pushes the result in the desired direction
- For special cases special penalties may be needed
- Example 1: a non-negative impulse response
- Example 2: harmonic smoothing
- Example 3: monotone smoothing
Impulse response

- Consider special “data”
- All zeros, but one 1 (an impulse)
- The result of smoothing we call the impulse response
- It shows how data get “smeared out”
- For $d = 2$, it has negative side lobes
- This might not be desirable
- Solution: use penalty $\lambda^2||D_2z||^2 + 2\lambda||D_1z||^2$
- Here $D_1$ ($D_2$) forms first (second) differences

Harmonic smoothing

- Assume periodic data, period $p$
- Wanted: smooth limit that approaches (co)sine
- Solution: a specialized penalty
  \[
  R = \sum_i (z_i - 2\phi z_{i-1} + z_{i-2})^2
  \]
- Where $\phi = \cos(2\pi/p)$

Illustration of positive impulse response

Illustration of harmonic smoothing
Varying penalties

- Our penalties had the same weight everywhere
- But we can change that:
  \[ R = \lambda \sum_i v_i (\Delta d z_i)^2 \]
- Or, with \( V = \text{diag}(v) \), \( R = z' D' V D z \)
- New problem: how to choose \( v \)?
- Simple choice: \( v_i = \exp(\gamma i) \)
- Optimize \( \lambda \) and \( \gamma \)

Asymmetric penalties and monotone smoothing

- Sometimes we want a smooth increasing result
- Smoothing alone does not guarantee a monotone shape
- We need a little help
- Additional asymmetric penalty \( P = \kappa \sum_i v_i (z_i - z_{i-1})^2 \)
- With \( v_i = 1 \) if \( z_i < z_{i-1} \) and \( v_i = 0 \) otherwise
- The penalty only works where monotonicity is violated
- With large \( \kappa \) we get the desired result
- This idea also works for convex smoothing
Example of monotone smoothing

Wrap-up

- The discrete smoother is simple and powerful
- It can be used for normal and non-normal data
- Penalty pushes solution in desired direction
- Penalty fills gaps in the data
- Desirable limits: polynomial or (co)sine
- “Designer penalties” open up new terrain
- Data have to be equally spaced (but gaps are allowed)
- Next session: the real thing, combination with B-splines
Part 2

The splendor of splines
Basis functions for polynomial curve fit

- Regression model $\mu = X\alpha$
- Columns of matrix $X$: basis functions. Polynomial basis

The motorcycle data

- Simulated crash experiment, a classic in smoothing
- Acceleration of motorcycle helmets measured

Polynomial fit to motorcycle data

- High degree (here 9) needed for decent curve fit
- Bad numerical condition (use orthogonal polynomials)
Sensitivity to data changes

- Longer left part (near zero)
- Notice the wiggles, also at the right

Motorcycle helmet data, polynomial of degree 9

The trouble with polynomials

- High degree (10 or more) may be needed
- Basis functions (powers of $x$) are global
- Moving one end (vertically) moves the other end too
- Good fit at one end spoils it at the other end
- Unexpected, but unavoidable, wiggles
- The higher the degree the more sensitive
- Polynomials are not a great choice
- We switch to B-splines

One linear B-spline

- Two pieces, each a straight line, everything else zero
- Nicely connected at knots ($t_1$ to $t_3$) same value
- Slope jumps at knots

One quadratic B-spline

- Three pieces, each a quadratic segment, rest zero
- Nicely connected at knots ($t_1$ to $t_4$): same values and slopes
- Shape similar to Gaussian
One cubic B-spline

• Four pieces, each a cubic segment, rest zero
• At knots \( t_1 \) to \( t_5 \): same values, first & second derivatives
• Shape more similar to Gaussian

B-spline basis

• Basis matrix \( B \)
• Columns are B-splines

\[
\begin{bmatrix}
B_1(x_1) & B_2(x_1) & B_3(x_1) & \cdots & B_n(x_1) \\
B_1(x_2) & B_2(x_2) & B_3(x_2) & \cdots & B_n(x_2) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
B_1(x_m) & B_2(x_m) & B_3(x_m) & \cdots & B_n(x_m)
\end{bmatrix}
\]

• In each row only a few non-zero elements (degree plus one)
• Only a few basis functions contribute to \( \mu_i = \sum b_{ij} \alpha_j = B'_i \alpha \)
• (Software demo: PSPlay_bsplines)

Plot from PSPlay_bsplines program

A set of cubic B-splines

A B-spline basis
B-splines fit to motorcycle data

How to compute B-splines

• You can work from first principles
• Compute parameters of the polynomial segments
• Nine (3 times 3) coefficients, 8 constraints, height arbitrary
• Easier: recursive formula De Boor
• Even more easy: differences of truncated power functions (TPF)
• TPF: \( f(x|t,p) = (x - t)_+^p = (x - t)^p I(x > t) \)
• Power function when \( x > t \), otherwise 0
• Avoids bad numerical condition of TPF (De Boor)
**B-spline summary**

- B-splines are local functions, look like Gaussian
- B-splines are columns of basis matrix $B$
- Scaling and summing gives fitted values: $\mu = B\alpha$
- The knots determine the B-spline basis
- Polynomial pieces make up B-splines, join at knots
- General patterns of knots are possible
- But we only consider equal spacing
- Number of knots determines width and number of B-splines

**Technical details of P-splines**

- Minimize (with basis $B$)
  \[ Q = \|y - B\alpha\|^2 + \lambda \|D\alpha\|^2 \]
- Explicit solution:
  \[ \hat{\alpha} = (B' B + \lambda D'D)^{-1} B' y \]
- Hat matrix $H = (B' B + \lambda D'D)^{-1} B'$
- For a nice curve, compute $B^*$ on nice grid $x^*$
- Plot $B^*\hat{\alpha}$ vs $x^*$

**P-splines on one slide**

- Do regression on (cubic) B-splines
- Use equally spaced knots
- Take a large number of them (10, 20, 50)
- Put a difference penalty (order 2 or 3) on the coefficients
- Tune smoothness with $\lambda$ (penalty weight)
- Don’t try to optimize the number of B-splines
- Relatively small system of equations (10, 20, 50)
- Arbitrary distribution of $x$ allowed

**Properties of P-splines**

- Penalty $\sum_j (\Delta d\alpha_j)^2$
- Limit for strong smoothing is a polynomial of degree $d - 1$
- Interpolation: polynomial of degree $2d - 1$
- Extrapolation: polynomial of degree $d - 1$
- Conservation of moments of degree up to $d - 1$
- Many more B-splines than observations allowed
- The penalty does the work!
- (Software demo: PSPlay.psplines)
Motorcycle helmet data

Optimal P-spline fit based on CVSEP

Standard errors

- Sandwich estimator
  \[ \text{var}(\hat{y}) = \text{var}(Hy) \]
  \[ = H \text{var}(y) H' \]
  \[ \approx \sigma^2 \frac{B(B'B + \lambda D_d D_d)'B'(B'B + \lambda D_d D_d)^{-1}B'}{H} \]

- Use sqrt of diagonal, \( \hat{\alpha} \) approx. normal, \( \hat{y} \pm 2\text{se}(\hat{y}) \)
- Again, effective model dimension: \( \text{tr}(H) \)
- Variance estimate
  \[ \hat{\sigma}^2 = \frac{|y - \hat{y}|^2}{m - \text{tr}(H)} \]

Optimal P-spline fit with twice se bands
**Generalized linear smoothing**

- It is just like a GLM (generalized linear model)
- With the penalty sneaked in
- Poisson example for counts \( y \)
- Linear predictor \( \eta = B\alpha \), expectations \( \mu = e^{\eta} \)
- Assumption \( y_i \sim \text{Pois}(\mu_i) \) (independent)
- From penalized Poisson log-likelihood follows iteration with
  \[
  (B'\bar{M}B + \lambda D'D)\alpha = B'(y - \bar{\mu} + \bar{M}B\bar{\alpha})
  \]
- Here \( M = \text{diag}(\mu) \)

**Introducing variances**

- Rewrite the penalized least squares goal:
  \[
  Q = \frac{||y - B\alpha||^2}{\sigma^2} + \frac{||D\alpha||^2}{\tau^2}
  \]
- Variance \( \sigma^2 \) of noise \( e \) in \( y = B\alpha + e \)
- Variance \( \tau^2 \) of contrast \( D\alpha \)
- First term: log of density of \( y \), conditional on \( \alpha \)
- Second term: log of (prior) density of \( D\alpha \)
- So \( \lambda \) is a ratio of variances: \( \lambda = \sigma^2/\tau^2 \)

**Alternative interpretations of penalties**

- Consider penalized least squares: minimize
  \[
  Q = ||y - B\alpha||^2 + \lambda||D\alpha||^2
  \]
- That penalty is rather useful
- But it seems to come out of the blue
- Can we connect it to established models?
- Yes: Bayes, or mixed models

**Bayesian simulation**

- We look for posterior distributions of \( \alpha, \sigma^2, \tau^2 \)
- Use Gibbs sampling
- “Draw” \( \alpha \) conditional on \( \sigma^2 \) and \( \tau^2 \)
- “Draw” \( \sigma^2 \) and \( \tau^2 \), conditional on \( \alpha \)
- These are relatively simple subproblems
- Repeat many times, summarize results
Sketch of Bayesian P-splines MCMC steps

# Prepare some useful summaries
BB = t(B) %*% B; By = t(B) %*% y; yy = t(y) %*% y; P = t(D) %*% D

# Run a Markov chain (loop not shown):

# Update coefficients
U = BB / sig2 + P / tau2
Ch = chol(U)
a0 = solve(Ch, solve(t(Ch), By)) / sig2;
a = solve(Ch, rnorm(length(a0))) + a0;

# Update error variance
r2 = yy - 2 * t(a) %*% By + t(a) %*% BB %*% a;
sig2 = as.single(r2 / rchisq(1, m));

# Update roughness variance
r = D %*% a;
tau2 = as.single(t(r) %*% r / rchisq(1, nb - 2));

Pros and cons of Bayesian P-splines

- You fit P-splines thousand of times: much work
- But all uncertainties are quantified
- This not the case when optimizing AIC, CV
- Theory applies to non-normal smoothing too
- But simulations (of $\alpha$) are much harder
- Metropolis-Hastings: acceptance rates need tuning
- More on this: Lang et al.: papers, program BayesX
- More modern tools: Langevin sampler, INLA
Mixed model

- See penalty as log of “mixing” distribution of $D\alpha$
- Mixed model software is good at estimating variances
- $D\alpha$ has singular distribution, rewrite the model
- Introduce “fixed” part $X$ and “random” part $Z$
- $y = B\alpha = X\alpha + Za$, with $Z = BD'(DD')^{-1}$
- And $X$ containing powers of $x$ up to $d − 1$
- Now a well behaved: independent components

Mixed model for P-splines in R

```r
# Based on work by Matt Wand
# Compute fixed (X) and mixed (Z) basis
B = bbase(x, 0, 1, 10, 3)
n = dim(B)[2]
d = 2;
D = diff(diag(n), differences = d)
Q = solve(D %*% t(D), D);
X = outer(x, 0:(d - 1), 'ˆ');
Z = B %*% t(Q)

# Fit mixed model
lmf = lme(y ~ X - 1, random = pdIdent(~ Z - 1))
beta.fix <- lmf$coef$fixed
beta.mix <- unlist(lmf$coef$random)
```

Example of P-spline fit with mixed model

EM-type algorithm for P-spline mixed model

- Deviance
  $$-2l = m \log \sigma + n \log \tau + ||y - B\alpha||^2/\sigma^2 + ||D\alpha||^2/\tau^2$$
- ML solution ($\lambda = \sigma^2/\tau^2$)
  $$(B'B + \lambda D'D)\hat{\alpha} = B'y$$
- One can prove (ED is effective dimension):
  $$E(||y - B\hat{\alpha}||^2) = (m - ED)\sigma^2; \quad E(||D\hat{\alpha}||^2) = ED\tau^2$$
- Use these to estimate $\hat{\sigma}^2$ and $\hat{\tau}^2$ from fit
- Refit with $\lambda = \sigma^2/\tau^2$, repeat
Example of P-spline fit with EM

Handling a penalty by data augmentation

\[ Q = \|y - B\alpha\|^2 + \lambda\|D\alpha\|^2 \]

- Solve linear system

\[ (B'B + \lambda D'D)\alpha = B'y \]

- Equivalent: regression with augmented data:

\[ B_+ = \begin{bmatrix} B \\ \lambda D \end{bmatrix}; \quad y_+ = \begin{bmatrix} y \\ 0 \end{bmatrix}; \]

P-splines with \(L_1\) (P1-splines)

- \(L_1\) norm: sum of absolute values
- \(L_1\) regression on B-spline basis \(B(x)\), with \(L_1\) difference penalty

\[ Q = |y - B\alpha| + \lambda|D\alpha| \]

- Equivalent data augmentation:

\[ B_+ = \begin{bmatrix} B \\ \lambda D \end{bmatrix}; \quad y_+ = \begin{bmatrix} y \\ 0 \end{bmatrix}; \]

- Solve with linear programming
- Use \texttt{l1fit()} or \texttt{rq()} (package \texttt{quantreg}) in R

P1-splines are robust
Generalized additive models

- One-dimensional smooth model: \( \eta = f(x) \)
- Two-dimensional smooth model: \( \eta = f(x_1, x_2) \)
- General \( f \): any interaction between \( x_1 \) and \( x_2 \) allowed
- We want to avoid two-dimensional smoothing
- Generalized additive model: \( \eta = f_1(x_1) + f_2(x_2) \)
- Both \( f_1 \) and \( f_2 \) smooth (Hastie and Tibshirani, 1990)
- Higher dimensions straightforward

The old way: backfitting for GAM

- Assume linear model: \( E(y) = \mu = f_1(x_1) + f_2(x_2) \)
- Assume: approximations \( \hat{f}_1 \) and \( \hat{f}_2 \) available
- Compute partial residuals \( r_1 = y - \hat{f}_2(x_2) \)
- Smooth scatterplot of \( (x_1, r_1) \) to get better \( \hat{f}_1 \)
- Compute partial residuals \( r_2 = y - \hat{f}_1(x_1) \)
- Smooth scatterplot of \( (x_2, r_2) \) to get better \( \hat{f}_2 \)
- Repeat to convergence

More on backfitting

- Start with \( \hat{f}_1 = 0 \) and \( \hat{f}_2 = 0 \)
- Generalized residuals and weights for non-normal data:
  - Any smoother can be used
  - Convergence can be proved, but may take many iterations
  - Convergence criteria should be strict

PGAM: GAM with P-splines

- Use B-splines: \( \eta = f_1(x_1) + f_2(x_2) = B_1\alpha_1 + B_2\alpha_2 \)
- Combine \( B_1 \) and \( B_2 \) to matrix, \( \alpha_1 \) and \( \alpha_2 \) to vector:
  \[
  \eta = [B_1 : B_2] \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = B'\alpha^* 
  \]
- Difference penalties on \( \alpha_1, \alpha_2 \), in block-diagonal matrix
- Penalized GLM as before: no backfitting
P-GAM fitting

- Maximize
  \[ l^* = l(\alpha; B, y) - \frac{1}{2} \lambda_1 |D_{d1} \alpha_1|^2 - \frac{1}{2} \lambda_2 |D_{d2} \alpha_2|^2 \]
- Iterative solution:
  \[ \hat{\alpha}_{t+1} = (B' \hat{W}_t B + P)^{-1} B' \hat{W}_t \hat{z}^* \]

where
\[
P = \begin{bmatrix}
  \lambda_1 D'_{d1} D_{d1} & 0 \\
  0 & \lambda_2 D'_{d2} D_{d2}
\end{bmatrix}
\]

PGAM advantages

- No backfitting, direct solution
- Fast computation
- Equations of moderate size, compact result (\(\alpha^*\))
- Explicit computation of hat matrix:
- Easy to compute CV, ED, AIC
- Easy standard errors
- No iterations, no convergence criteria to set
- Implemented in Simon Wood’s mgcv package

Features of P-spline GAMs

- \(ED = \text{trace}(\hat{H}) = \text{trace}(B(B' \hat{WB} + P)^{-1} B' \hat{W})\)
- AIC = deviance\((y; \hat{\alpha}) + 2 \text{trace}(\hat{H})\)
- Standard error of \(j\)th smooth
  \[ B_j (B' \hat{WB} + P)^{-1} B' \hat{WB} (B' \hat{WB} + P)^{-1} B'_j \]
- GLM diagnostics accessible
- Easy combination with additional linear regressors/factors
- Example: \([B_1 : B_2 : X]\) (no penalty on \(X\) coefficients)

Two-dimensional smoothing with P-splines

- Use tensor product B-splines: \(T_{jk}(x, y) = B_j(x) \tilde{B}_k(y)\)
- Equally spaced knots on 2D grid
- Matrix of coefficients \(A = [\alpha_{jk}]\)
- Difference penalties on coefficients
- Penalties on rows/columns of \(A\)
Implementation of the basis

- Model contains matrix of coefficients $A$
- Transform to vector: $\alpha = \text{vec}(A)$
- Kronecker product of bases
  \[ T = B_1 \otimes B_2 \]
- $T$ is of dimension $m \times (n\bar{n})$

Two-dimensional penalized estimation

- Objective function
  \[ Q_P = \text{RSS} + \text{Row Penalty} + \text{Column Penalty} \]
  \[ = \text{RSS} + \lambda_1 \sum_{j=1}^{n} A_j D_d^T D_d A_j^T + \lambda_2 \sum_{k=1}^{\bar{n}} A_k^T D_d^T D_d A_k \]
  \[ = |z - T\alpha|^2 + \lambda_1 |P_1\alpha|^2 + \lambda_2 |P_2\alpha|^2. \]
- Penalize rows of $A$ with $D_d$
- Penalize columns of $A$ with $D_{\bar{d}}$
- Number of equations is $n\bar{n}$
Details of row and column penalties

- Must also carefully arrange ("stack") penalties
- Block diagonal to break (e.g. row to row) linkages:
  - \( P_1 = D \otimes I_n \)
  - \( P_2 = I_n \otimes D \)

The ethanol data

- Nitrogen oxides in motor exhaust: \( NO_x \) (z)
- Compression ratio, C (x), equivalence ratio, E (y)

PGAM fit for ethanol data
PGAM components for ethanol data

2D smoothing of ethanol data

- Tensor products of cubic B-splines
- Dimension: 64 (8 by 8)
- Fit computed on 400 points
- Residuals (SD) reduced to 60%, compared to GAM

Tensor P-spline fit to ethanol data

Another 2D application

- Printed circuit board
- Clamping causes warping (approx. 0.5 mm)
- Laser inspection of deformation
- Input: 1127 observations
- Cubic P-spline tensor products: 13 by 13
- Interpolation at 1600 points
Printed circuit board data

Higher dimensions

- Triple (or higher) tensor products possible
- Difference penalty for each dimension
- Many equations: $n^3 (n^4)$
- Reduce number of B-splines
- Data generally sparse in more dimensions
- Special algorithm for (possibly incomplete) data on grids
- Speed-up 10 to 1000 times

Wrap-up

- P-splines are useful
- They are beautiful too
- People like them: many citations
- The penalties form the skeleton
- The B-splines put the flesh on it
- See back of handout for further reading

About software

- We (PE and BD) did not write a package
- Too busy exploring new applications ;-)  
- Some scripts on stat.lsu.edu/bmarx
- Simon Wood’s mgcv package offers a lot
- I’m always willing to help
- And to share my software
- p.eilers@erasmusmc.nl
Further reading


Ill-posed problems with counts, the composite link model and penalized likelihood. *Statistical Modelling* 7, 239–254.


Consumer score card for smoothers

This score card is reproduced from our paper in Statistical Science (1996).

<table>
<thead>
<tr>
<th>Aspect</th>
<th>KS</th>
<th>KSB</th>
<th>LR</th>
<th>LRB</th>
<th>SS</th>
<th>SSB</th>
<th>RSF</th>
<th>RSA</th>
<th>PS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed of fitting</td>
<td>—</td>
<td>+</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Speed of optimization</td>
<td>—</td>
<td>+</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Boundary effects</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>—</td>
</tr>
<tr>
<td>Sparse designs</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>+</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>+</td>
</tr>
<tr>
<td>Semi parametric models</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>—</td>
</tr>
<tr>
<td>Non-normal data</td>
<td>+</td>
<td>+</td>
<td>—</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>—</td>
</tr>
<tr>
<td>Easy implementation</td>
<td>+</td>
<td>—</td>
<td>+</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>—</td>
<td>—</td>
<td>+</td>
</tr>
<tr>
<td>Parametric limit</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Specialized limits</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>+</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Variance inflation</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>—</td>
</tr>
<tr>
<td>Adaptive flexibility possible</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>—</td>
<td>+</td>
<td>—</td>
<td>+</td>
<td>—</td>
</tr>
<tr>
<td>Adaptive flexibility available</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Compact result</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>—</td>
<td>+</td>
<td>—</td>
</tr>
<tr>
<td>Conservation of moments</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>—</td>
<td>+</td>
<td>—</td>
</tr>
<tr>
<td>Easy standard errors</td>
<td>—</td>
<td>—</td>
<td>+</td>
<td>—</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>—</td>
</tr>
</tbody>
</table>

Consumer test of smoothing methods. The abbreviations stand for

KS  kernel smoother
KSB kernel smoother with binning
LR  local regression
LRB local regression with binning
SS  smoothing splines
SSB smoothing splines with band solver
RSF regression splines with fixed knots
RSA regression splines with adaptive knots
PS  P-splines

The row “Adaptive flexibility available” means that a software implementation is readily available.