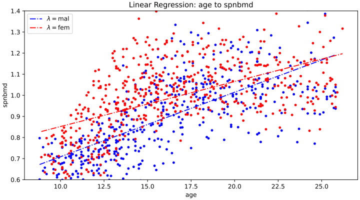
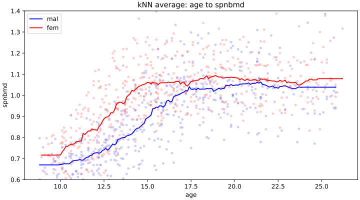
Piecewise Polynomials and Splines

- **Regression** We interpolate/predict a continuous variable
- Example: Bone Mineral Density for adolescent male/female





- Left: Linear regression
- Right: k-nearest neighbour estimate
- We may generalize the linear regression

  $\Rightarrow$ splines (piecewise polynomial regression)
- We may generalize the kNN

  $\Leftarrow$ kernel methods.

Basis expansion and regularization, Splines

Linear assumes linear function of X.

- Regression: We estimate $f(X) = \mathbb{E}(Y|X)$

Linear basis expansion in X

- we replace the vector of inputs X with additional variables h_m,
- $h_m(X) : \mathbb{R}^p \to \mathbb{R}$, $m = 1, \ldots, M$.

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

- 'the only change' is a different matrix of the features X, further fit is the same.
- Usually, we search $f_j(X_j)$ for each dimension by a backfitting algorithm in a **generalized additive model (GAM)**

$$\mathbb{E}(Y|X_1, \ldots, X_p) = \alpha + f_1(X_1) + \ldots + f_p(X_p)$$

  - where f_j's are unspecified smooth functions
  - X_j predictors, Y the outcome.
- For now, we consider one-dimensional feature X.
-

Simple derived features

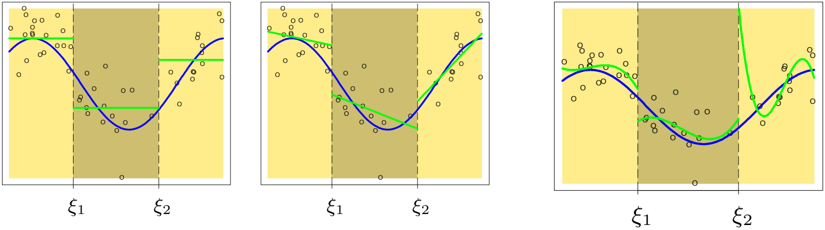
- We fit the model:

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

- $h_m(X) = X_m$, $m = 1, \ldots, M$ recovers the original linear model.
- $h_m(X) = X_j^2$ or $h_m(X) = X_j X_k$ polynomial terms to achieve higher-order Taylor expansions.
  - ! The number of variables grows exponentially in the degreee of the polynomial.
- $h_m(X) = log(X_j)$, $\sqrt{X_j}$, $||X||$,..., other nonlinear transformations.
- $h_m(X) = I(L_m \leq X_k < U_m)$, an indicator for a region of X_k.
  - piecewise constant contribution for X_K.
  - With non-overlapping regions used in regression trees.
- $h_m(X) = max((X_j - \xi_k)^3, 0)$ piecewise-polynomial spline basis
- wavelet bases.

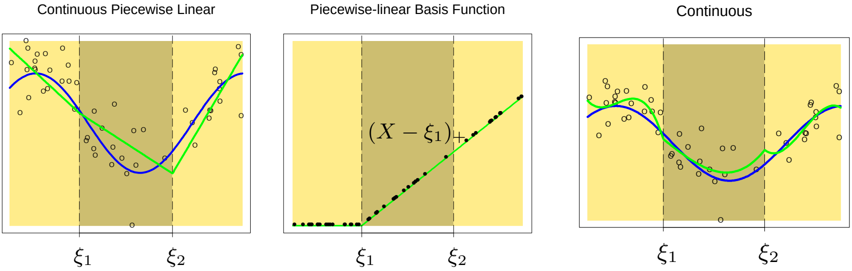
Piecewise Polynomials and Splines

- A **piecewise polynomial** function $f(X)$ is obtained by
  - division the domain of X into continuous intervals by the knots $\xi_1, \ldots, \xi_{M-1}$
  - and representing f by a separate polynomial in each interval.
  - Examples:
    - Three basis functions:
      $h_1(X) = I(X < \xi_1)$, $h_2(X) = I(\xi_1 \leq X < \xi_2)$, $h_3(X) = I(\xi_2 \leq X)$.
    - Additional linear functions:
      $h_{m+3} = h_m(X) \cdot X$, $m = 1, \ldots, 3$.
    - Additional cubic functions:
      $h_{m+6} = h_m(X) \cdot X^2$, $h_{m+9} = h_m(X) \cdot X^3$, $m = 1, \ldots, 3$.



Continuous functions

- We add the continuity restriction: the value in ξ_j is the unique.
- Continuous piecewise linear basis:
  $h_1(X) = 1,\ h_2(X) = X,\ h_3(X) = (X - \xi_1)_+,\ h_4(X) = (X - \xi_2)_+.$
- We have spared two parameters for two continuity conditions.

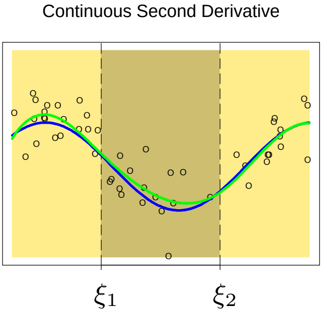


- For the cubic fit, the figure looks ugly, we need continous first and second derivative.

Cubic spline!

- **Cubic spline** is a piecewise cubic fit with continuous first and second derivatives at the knots ξ_i.

- The basis functions with knots ξ_1, ξ_2 are:
  $h_1(X) = 1$,
  $h_2(X) = X$,
  $h_3(X) = X^2$,
  $h_4(X) = X^3$,
  $h_5(X) = (X - \xi_1)^3_+$,
  $h_6(X) = (X - \xi_2)^3_+$.



- Parameter count:
  (3 regions)x(4 pars per region)-(2 knots)x(3 constraints per knot)=6.

Order-M splines

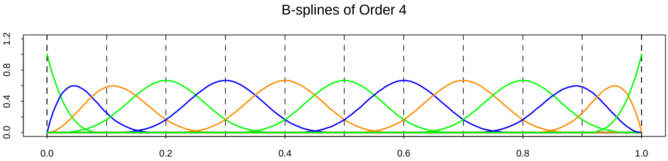
- Cubic spline is an order-4 spline.
- Generally, order-M spline with knots ξ_j, $j = 1, \ldots, K$ is a piecewise-polynomial of order $(M-1)$ and has continuous derivatives to order $(M-2)$.
- General truncated basis functions are:
  - $h_j(X) = X^{j-1}$, $j = 1, \ldots, M$,
  - $h_{M+\ell} = (X - \xi_\ell)_+^{M-1}$, $\ell = 1, \ldots, K$.
- **Regression splines**
  - splines with fixed knots
  - usually at percentiles of the data X.
  - the number of knots is specified by the degree an the degrees of freedom $(df - M)$. h_0 does not count.

B-splines

B-splines use other basis describing the same linear feature space.

- $\{h_i\}$ is a basis of a linear space of functions
- we may choose a different base to cover the same space of functions.
- B-splines are more stable numerically, useful for large number of knots K.
- B-splines have quite difficult recursive formula (not needed for the exam).

$$
\begin{aligned}
B_{i,1}(x) &= \begin{cases} 1 & \text{if } \xi_i \le x \le \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \\
B_{i,k+1}(x) &= \omega_{i,k}(x) B_{i,k}(x) + [1 - \omega_{i+1,k}(x)] B_{i+1,k}(x) \\
\omega_{i,k}(x) &= \begin{cases} \frac{x-\xi_i}{\xi_{i+k}-\xi_i} & \text{if } \xi_{i+k} \ne \xi_i \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$



Computational complexity

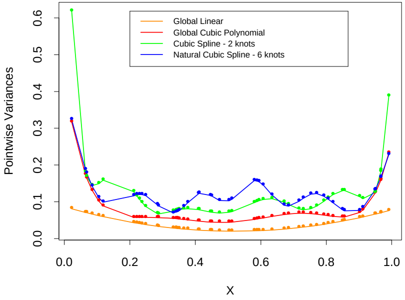
Spline fit time complexity

- (Standard) regression splines
  - N observations, $K + M$ variables (basis functions) take $O(N(K+M)^2 + (K+M)^3)$.
- B-splines
  - sort values of X
  - Cubic B splines have local support, B is lower 4-banded.
  - order $(M+1)$ B splines have local support, B is lower $(M+1)$-banded.
  - Cholesky decomposition $B = LL^T$ can be computed easily.
  - Solution of $\hat{f}$ is in $O(N(M+1))$ operations.

B-splines implemented in scipy and statmodels.

Natural Cubic Spline

- Polynomial fit tends to be erratic near the boundaries.



- **Natural cubic spline!** is a spline that the function is linear beyond the boundary knots.
- Basis functions N_i, $i = 1, \ldots, K$:
  $N_1(X) = 1, N_2(X) = X, N_{k+2}(X) = d_k(X) - d_{K-1}(X)$ for

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

Smoothing Splines

- **Smoothing Spline**! uses the maximal number of knots: N, the number of examples.
- But, we need a penalty for model complexity.

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

- λ is **smoothing parameter**
  - $\lambda = 0$: can be any function that interpolates the data.
  - $\lambda = \infty$: the simple least squares line fit, no nonzero second derivative is tolerated.
- Has a unique finite-dimensional minimizer, a natural cubic spline with knots at the unique values of the x_i, $i = 1, \ldots, N$.
- The solution is a **natural cubic spline**: $f(x) = \sum_{j=1}^{N} N_j(x)\theta_j$.
- The criterion reduces for:

$$RSS(\theta, \lambda) = (\mathbf{y} - \mathbf{N}\theta)^T(\mathbf{y} - \mathbf{N}\theta) + \lambda\theta^T\Omega_N\theta$$

- where $\{\mathbf{N}\}_{ij} = N_j(x_i)$ and $\{\Omega\}_{jk} = \int N_j''(t)N_k''(t)dt$.

Ω

let $a = x_1 = 0$, $b = x_{101} = 1$, and knots $\xi_l = x_{l+1}$ for $l = 1, \ldots, K$ and $K = 99$. Also, the basis functions for a cubic spline $M = 4$ are

$$\begin{aligned} h_j(x) &= x^{j-1} & j = 1, \ldots, M, \\ h_{M+l}(x) &= (x - \xi_l)_+^{M-1} & l = 1, \ldots, K. \end{aligned}$$

Then, $\boldsymbol{H} = (h_j(x_i))_{N,M+K}$ where $h_j(x_i)$ is for the i-th row and the j-th column. Let $\mathbf{\Omega} = (\omega_{i,j})_{M+K,M+K}$ be a symmetric matrix and the upper triangular $\omega_{i,j} = \int_a^b h_i''(t) h_j''(t) dt$ is

$$\begin{aligned} \omega_{i,j} &= 0 & \text{for } i < M, \\ \omega_{M,j} &= \tfrac{1}{3}b^3 - \tfrac{1}{2}b^2\xi_j + \tfrac{1}{6}\xi_j^3 & \text{for } j > M, and \\ \omega_{i,j} &= \tfrac{1}{3}(b^3 - \xi_0^3) - \tfrac{1}{2}(b^2 - \xi_0^2)(\xi_{i-M} + \xi_{j-M}) + (b - \xi_0)\xi_{i-M}\xi_{j-M} & \text{for } j \geq i > M, \end{aligned}$$

where $\xi_0 = \max\{\xi_{i-M}, \xi_{j-M}\}$.

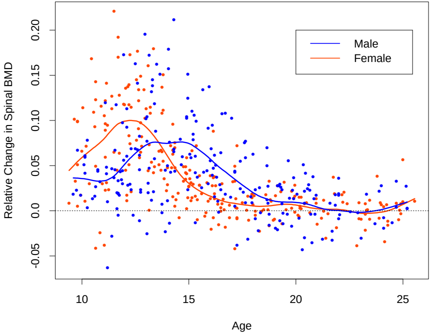
https://vardeman.public.iastate.edu/stat602/602x_hw4_sol.pdf

Smoothing Splines solution

- Smoothing spline solution is a generalized ridge regression
$$\hat{\theta} = (\mathbf{N}^T\mathbf{N} + \lambda\Omega_N)^{-1}\mathbf{N}^T\mathbf{y}$$
- The fitted smoothing spline is given by:
$$\hat{f}(x) = \sum_{j=1}^{N} N_j(x)\hat{\theta}_j$$

Example

- Bone mineral density (BMD) in adolescents.
- Response: the change in BMD over two consecutive visits, typically about one year apart.
- coded by gender, females precedes growth spurt about two years.
- $\lambda \approx 0.00022$, $df_\lambda = 12$.



Degrees of Freedom and Smoother Matrices

- Smoothing spline is a linear smoother:

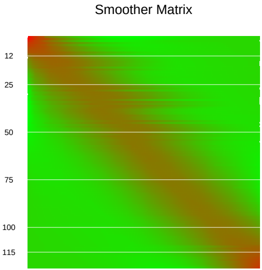
$$\begin{aligned}\hat{f} &= \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\Omega_N)^{-1}\mathbf{N}^T\mathbf{y} \\ &= \mathbf{S}_\lambda \mathbf{y}\end{aligned}$$

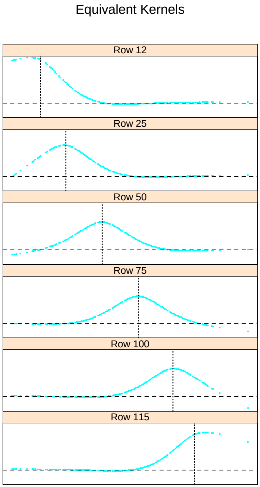
- $\mathbf{S}_\lambda$ is known as smoother matrix.
- $df_\lambda = trace(\mathbf{S}_\lambda)$
  - the sum of the diagonal elements
  - $\lambda \approx 0.00022$ derived numerically by solving $trace(\mathbf{S}_\lambda) = 12$.

from scipy.interpolate import make_smoothing_spline

Smoother Matrix

- rows $\mathbf{S}_\lambda$ ordered with x
- right: selected rows
- $\lambda \to 0$ means $df_\lambda \to N$ and $\mathbf{S}_\lambda \to \mathbf{I}$
- $\lambda \to \infty$ means $df_\lambda \to 2$ and $\mathbf{S}_\lambda \to \mathbf{H}$, the hat matrix for linear regression on $\mathbf{x}$.
- $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ since $(\hat{y} = Hy)$





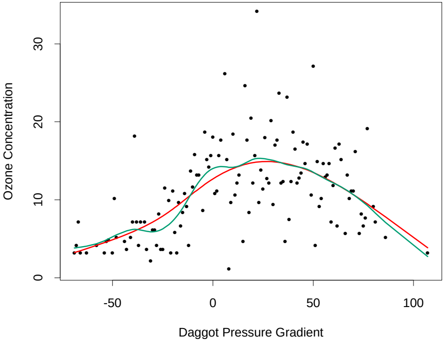
Pollution data example

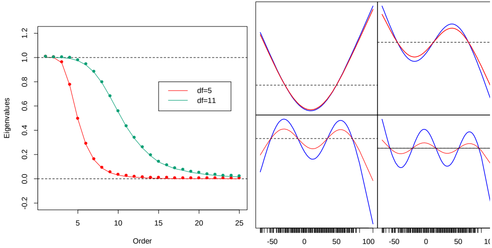
Example

- 128 observations of pressure and ozone.
- Two fitted smoothing splines.
- third to sixth eigenvectors of the spline smoother matrices u_k against x.
- eigendecomposition of S: eigenvaules d_k (right)

$$\mathbf{S}_\lambda = \sum_{k=1}^{N} \rho_k(\lambda) u_k u_k^T$$

- $\rho_k(\lambda) = \frac{1}{1+\lambda d_k}$.
- Right: 3rd-6th eigenvectors as a function of x and smoothed $df = 5$ in red.



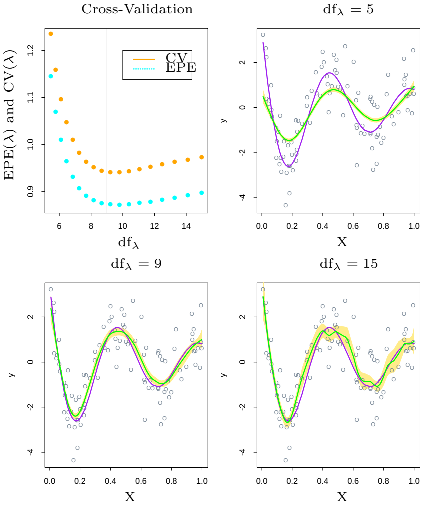


Selection degrees of freedom

- The degrees of freedom *df* (or the complexity penalty λ) are usually selected to minimize the expected prediction error.
- More specifically, the crossvalidation estimate of the error.

Example

- $f(X) = \frac{sin(12(X+0.2))}{X+0.2}$
- $Y = f(X) + \epsilon$
- $X \sim U[0,1]$, $\epsilon \sim N(0,1)$, $N = 100$.
- *df* selected by crossvalidation is 9.



Multidimensional Splines

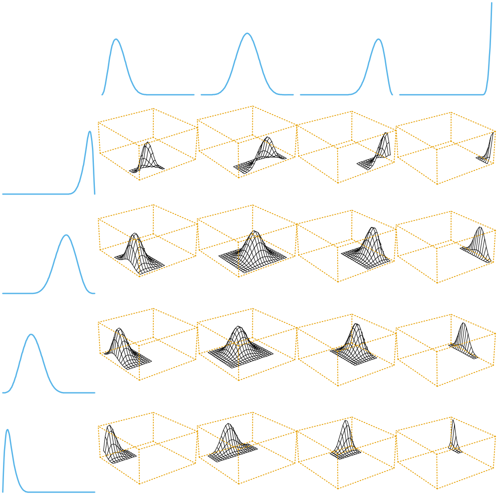
- $X \in \mathbb{R}^2$
- $h_{1k}(X_1),\ k = 1, \ldots, M_1$ in the first coordinate
- $h_{2k}(X_2),\ k = 1, \ldots, M_2$ in the second coordinate.
- $M_1 \times M_2$ dimensional tensor product basis is defined by

$$g_{jk}(X) = h_{1j}(X_1)h_{2k}(X_2)$$

- can be used for representing a two-dimensional function:

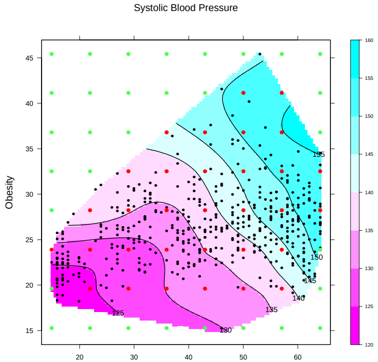
$$g(X) = \sum_{j=1}^{M_1} \sum_{k=1}^{M_2} \theta_{jk} g_{jk}(X)$$

- coefficients can be fitted by least squares.



Multidimensional smoothing splines

- Let us place the knot into each example
- and add a complexity penalty J (below).
- It can be generalized for an arbitrary dimension.
- The solution has the form:
  - $f(x) = \beta_0 + \beta^T x + \sum_{j=1}^{N} \alpha_j h_j(x)$
  - where $h_j(x) = \eta(||x - x_j||)$ and $\eta(z) = z^2 2 \log z^2$.



- complexity $O(N^3)$
- or $O(NK^2 + K^3)$ with K knots.

$$J[f] = \int \int_{\mathbb{R}^2} \left[\left(\frac{\partial^2 f(x)}{\partial x_1^2} \right) + \left(\frac{\partial^2 f(x)}{\partial x_1 \partial x_2} \right) + \left(\frac{\partial^2 f(x)}{\partial x_2^2} \right) \right] dx_1 dx_2$$

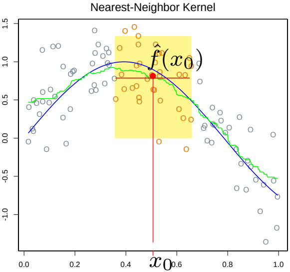
implemented in R and OpenCV

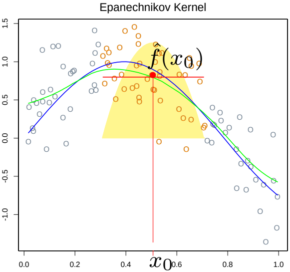
Summary

We learned about

- regression **splines** (one dimensional X) - these formulas you should know
- **B-splines** - for faster fit - no formulas necessary
- **natural splines** - linear on the borders
- **smoothing splines** - complexity penalty for the second derivative
  - the solution is a natural spline.
- Generalizations to more dimensions
  - thin plate splines
  - multidimensional smoothing splines.

- estimate regression function $f(x) \in \mathbb{R}$
- a different but simple model separately at each query point x_0.
- The resulting $\hat{f}(X)$ is smooth in $\mathbb{R}^p$.
- Localization is achieved via a weighting function er **kernel** $k_\lambda(x_0, x_i)$
  - assigns a weight to x_i based on its distance form x_0.
- λ is a parameter that dictates the width of the neighbourhood.
- **memory based methods**
  - little or no training
  - the model is the entire training data set.





Kernel Smoothing Methods

- **k-Nearest Neighbour** kernel
  - $N_k(x)$ is the set of k points nearest to x in squared distance
  - all have equal weight
  - $\hat{f}(x) = \frac{1}{k}\sum_{x_i \in N_k(x)} y_i$.
  - $\hat{f}(x)$ is bumpy, discontinuous.
- **Nadaraya-Watson** kernel-weighted average

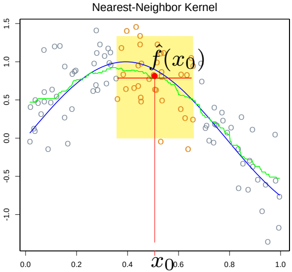
$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} k_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} k_\lambda(x_0, x_i)}$$ sure

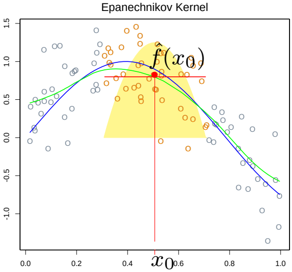
- with the **Epanechnikov** quadratic kernel

$$k_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right)$$

- with

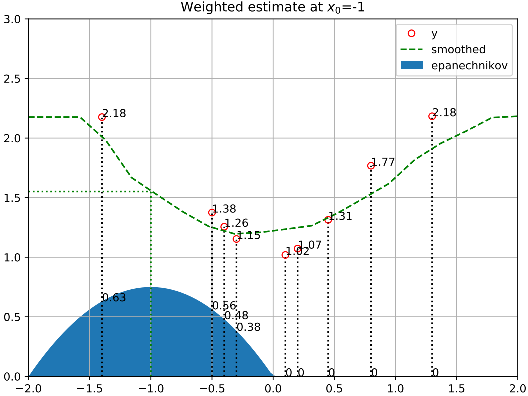
$$D(t) = \begin{cases} \frac{3}{4}(1 - t^2) & \text{if } |t| \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$





Example!

- Red circles: data
- Blue: Epanechnikov kernel for (-1.0)
- Predicted values: green dashed line
- predicted value $\hat{f}(-1.0) = 1.55$.



$$\frac{(0.63 * 2.18 + 0.56 * 1.38 + 0.48 * 1.26 + 0.38 * 1.15 + 0 * others)}{(0.63 + 0.56 + 0.48 + 0.38)} = 1.55$$

Kernels - variable width, shapes

- The width λ may vary $h_\lambda(x_0)$ with x_0
- mo general formula for he kernel

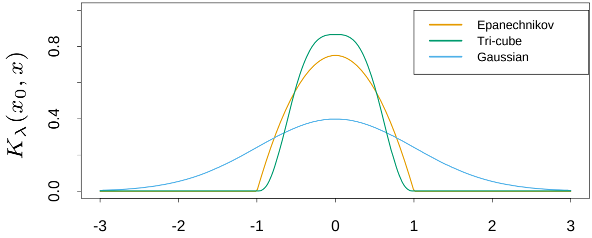
  $k_\lambda(x_0, x) = D\left(\frac{|x-x_0|}{h_\lambda(x_0)}\right)$
- for k-NN, $k_k(x_0, x) = \delta_{x \in Neighbours_k(x)}$
- where $x_{|k|}$ is the kth closest x_i to x_0.

- **Tri-cube kernel**

  $$D(t) = \begin{cases} (1 - |t|^3)^3 & \text{if } |t| \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$
- **Gaussian kernel**

  $$D(t) = \frac{1}{\lambda} e^{-\frac{\|x-x_0\|^2}{2\lambda}}$$
- **Epanechnikov**

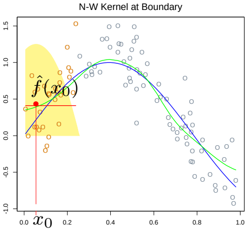
  $$D(t) = \begin{cases} \frac{3}{4}(1 - t^2) & \text{if } |t| \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$



Local Linear Regression

- Locally-weighted averages can be badly biased on the boundaries of the domain
- or whenever X are not equally spaced.
- Fitting straight lines may help (a bit).
- **Locally weighted regression**

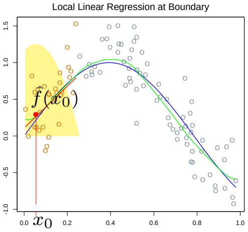
$$min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} k_\lambda(x_0, x_i)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2$$



- The estimate is: $\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$.
  - For $x^T \to (1, x)$, $\mathbf{X}$ is $N \times (p+1)$ matrix, $\mathbf{W}$ $N \times N$ diagonal matrix $k_\lambda(x_0, x_i)$. Then

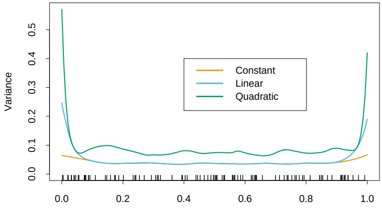
$$\hat{f}(x_0) = x_0^T (X^T W(x_0) X)^{-1} X^T) W(x_0) y$$

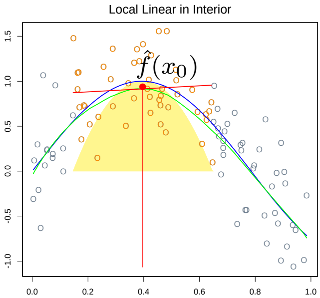
- what is linear function of y.

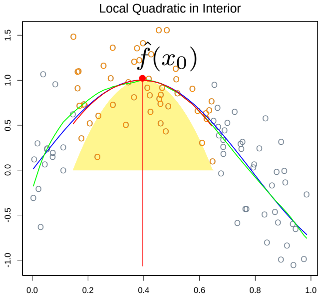


Local Polynomial Regression

- Local linear fits can help bias dramatically at the boundaries.
- local quadratic fits tend to be most helpful in reducing bias due to curvature in the interior of the domain.
- Recommended to select the degree by the application, not to combine linear boundaries and quadratic interior.

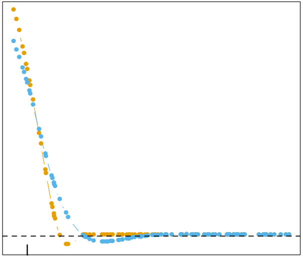


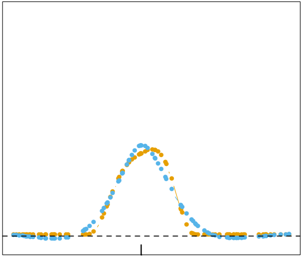




Selecting the Width of the Kernel

- crossvalidation
- $\hat{f} = S_\lambda y$
  - $df = trace(S_\lambda)$
- Right: comparison of the tri-cube local linear regression kernels (orange) and smoothing splines (blue) with matching degrees of freedom 5.86.



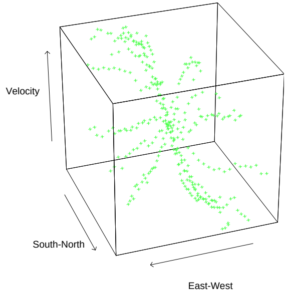


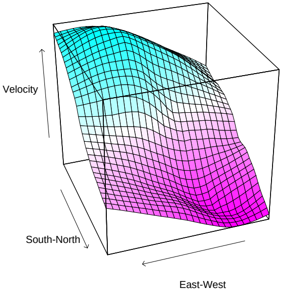
(Structured Local Regression in $\mathbb{R}^p$)

$$k_\lambda(x_0, x) = D\left(\frac{\|x - x_0\|}{h_\lambda(x_0)}\right)$$

- Structured local regression: a positive semidefinite matrix A to weigh the different coordinates:

$$k_\lambda(x_0, x) = D\left(\frac{(x - x_0)^T A (x - x_0)}{h_\lambda(x_0)}\right)$$





Computational Consideration

Kernel smoothing complexity

- Model is the entire training data set.
- The fitting is done at evaluation or prediction.
- Single observation x_0 fit is $O(N)$,
- expansion in M basis functions $O(M)$ for one evaluation, typically $M \sim O(logN)$.
- Basis function method have an initial cost at least $O(NM^2 + M^3)$.
- Smoothing parameter λ usually determined off-line by cross-validation, at cost of $O(N^2)$.
- Popular implementations of local regression *loess* is S-PLUS compute the fit exactly at M locations $O(NM)$ and interpolate to fit elsewhere ($O(M)$ per evaluation).

List of topics

1. Linear, ridge, lasso regression, k-neares neighbours,(formulas) overfitting, curse of dimensionality, (LARS)
2. Splines - the base, natural splines, smoothing splines; kernel smoothing: kernel average, Epanechnikov kernel.
3. Logistic regression, Linear discriminant analysis, generalized additive models
4. Train/test error and data split, square error, 0-1, crossentropy, AIC, BIC,(formulas) crossvalidation, one-leave-out CV, wrong estimate example
5. decision trees, information gain/entropy/gini, CART prunning,(formulas)
6. random forest (+bagging), OOB error, Variable importance, boosting (Adaboost(formulas) and gradient boosting), stacking, MARS,
7. Bayesian learning: MAP, ML hypothesis (formulas), Bayesian optimal prediction, EM algorithm
8. Clustering: k-means, Silhouette, k-medoids, hierarchical
9. Apriori algorithm, Association rules, support, confidence, lift
10. Inductive logic programming basic: hypothesis space search, background knowledge, necessity, sufficiency and consistency of a hypothesis, Aleph
11. Undirected graphical models, Graphical Lasso procedure, deviance, MRF
12. Gaussian processes: estimation of the function and its variance (figures, ideas).

List of topics

1. Linear, ridge, lasso regression, k-neares neighbours,(formulas) overfitting, curse of dimensionality, (LARS)
2. Splines - the base, natural splines, smoothing splines; kernel smoothing: kernel average, Epanechnikov kernel.
3. Logistic regression, Linear discriminant analysis, generalized additive models
4. Train/test error and data split, square error, 0-1, crossentropy, AIC, BIC,(formulas) crossvalidation, one-leave-out CV, wrong estimate example
5. decision trees, information gain/entropy/gini, CART prunning,(formulas)
6. random forest (+bagging), OOB error, Variable importance, boosting (Adaboost(formulas) and gradient boosting), stacking, MARS,
7. Bayesian learning: MAP, ML hypothesis (formulas), Bayesian optimal prediction, EM algorithm
8. Clustering: k-means, Silhouette, k-medoids, hierarchical
9. Apriori algorithm, Association rules, support, confidence, lift
10. Inductive logic programming basic: hypothesis space search, background knowledge, necessity, sufficiency and consistency of a hypothesis, Aleph
11. Undirected graphical models, Graphical Lasso procedure, deviance, MRF
12. Gaussian processes: estimation of the function and its variance (figures, ideas).

Table of Contens