# Statistics for Data Science: Week 13

Myrto Limnios and Rajita Chandak

Institute of Mathematics – EPFL

rajita.chandak@epfl.ch, myrto.limnios@epfl.ch

# Nonparametric Regression

So far we have discussed the following setup:

$$Y_i \mid \boldsymbol{x}_i^\top \overset{ind}{\sim} \text{Dist}[\phi_i] \rightarrow \begin{cases} \phi_i = g(\boldsymbol{x}_i^\top) = \boldsymbol{x}_i^\top \boldsymbol{\beta}, \\ \boldsymbol{\beta} \in \mathbb{R}^p, \end{cases}$$

with $\boldsymbol{\beta}$ to be estimated from data, e.g.

- Dist $= \mathcal{N}(\mu_i, \sigma^2)$ and $\mu_i = \boldsymbol{x}_i^\top \boldsymbol{\beta}$ (Gaussian linear regression)
- Dist $\in \text{ExpFamily}(\phi_i)$ and $\phi_i = \boldsymbol{x}_i^\top \boldsymbol{\beta}$ (GLM)

Why this is still a class of models we study?

- Representing the distribution as linearly dependent on the covariates is convenient and yields a very easy interpretation (both if $Y$ is continuous (regression) or discrete (classification))
- It is sometimes very useful when the sample size is very small (n)

Would now like to extend the model to a more flexible dependence - we want to replace the linear dependence on $\boldsymbol{x}$ with transformations of $\boldsymbol{x}$:

$$Y_i \mid \boldsymbol{x}_i^\top \overset{ind}{\sim} \text{Dist}[\phi_i] \to \begin{cases} \phi_i = g(\boldsymbol{x}_i^\top), \\ g \in \mathcal{F} \subset L^2(\mathbb{R}^p) \text{ (say)}, \end{cases}$$

with $g : \mathbb{R}^p \to \mathbb{R}$ unknown, to be estimated given data $\{(Y_i, \boldsymbol{x}_i^\top)\}_{i=1}^n$.

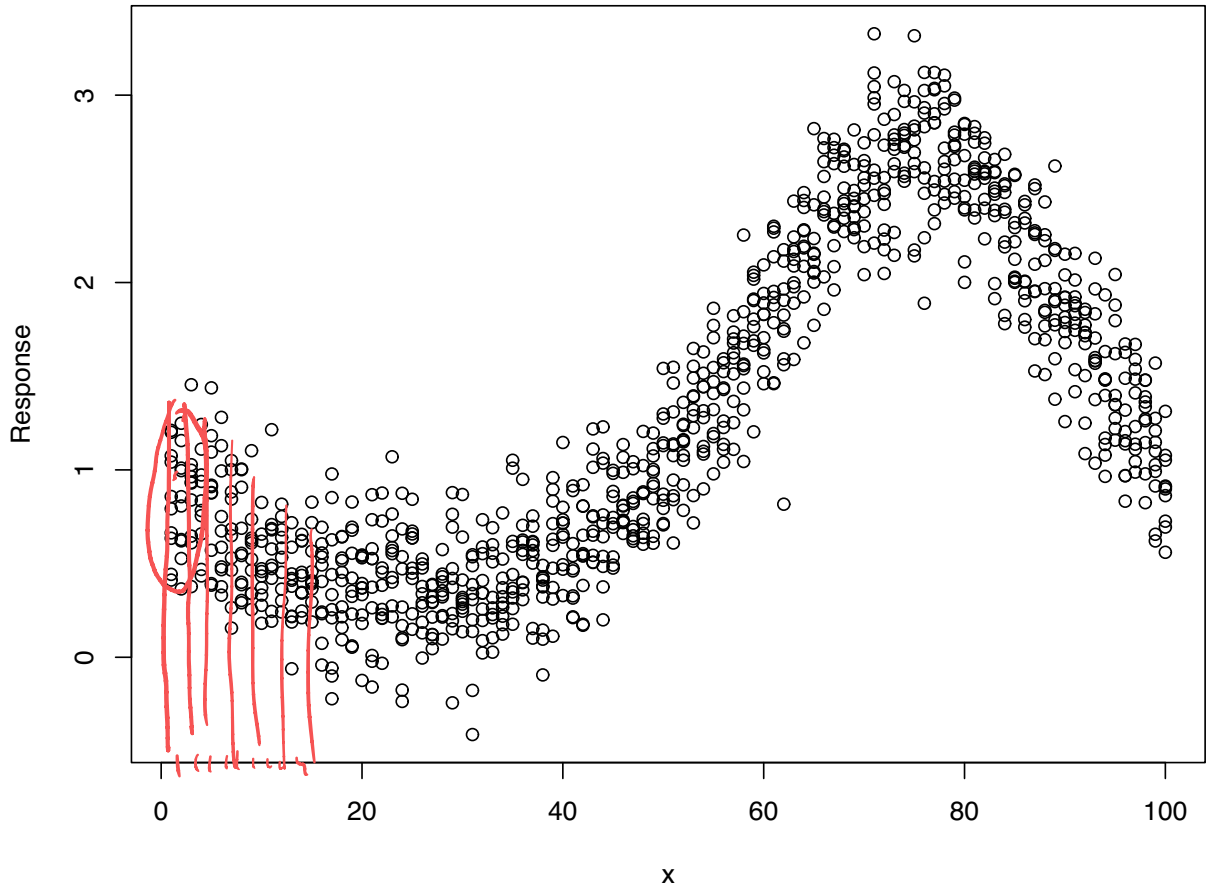
- A *nonparametric* problem (parameter $\infty$-dimensional)!
- How to estimate $g$ in this context?
- $\mathcal{F}$ is usually assumed to be a class of smooth functions (e.g., $C^k$).

Start from simplest problem:

$$\left.\begin{array}{l} \text{Dist} \equiv \mathcal{N}(\mu_i, \sigma^2) \\ x_i \in \mathbb{R} \end{array}\right\} \implies Y_i = g(x_i) + \varepsilon_i, \quad \varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$$

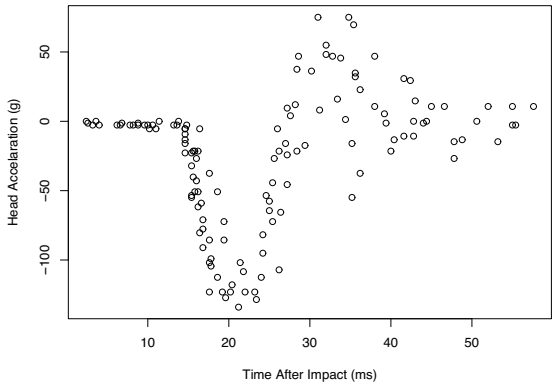
And ideally we would like to *linearly* decompose $g$ over a *dictionary* depending on a number of transformations of $\boldsymbol{x}$

- Ideally: large sample plus multiple $x_i$ with same value (many large covariate classes):

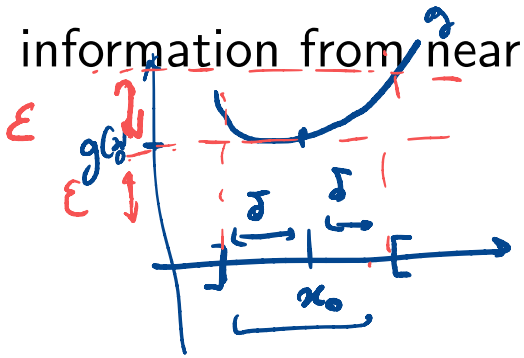



- Then average $Y_i$'s at each covariate class and interpolate . . .
- But this is never the case in practice... . . .

- Usually unique $x_i$ distinct:




- Here is where the smoothness assumption comes in
- Since have distinct value for each $x_i$, need to borrow information from nearby ...
- ... use continuity!!! (or even better, *smoothness*)




- Recall: A function $g : \mathbb{R} \to \mathbb{R}$ is *continuous* if:

$$\forall\ \epsilon > 0\ \exists\ \delta > 0 :\ |x - x_0| < \delta \implies |g(x) - g(x_0)| < \epsilon.$$

- So maybe average $Y_i$'s corresponding to $x_i$'s in a $\delta$-neighbourhood of $x$.
- Motivates the use of a kernel smoother ...

Naive idea: $\hat{g}(x_0)$ should be the average of $Y_i$-values with $x_i$'s "close" to $x_0$.

$$\hat{g}(x_0) = \frac{1}{\sum_{i=1}^n \mathbf{1}\{|x_i - x_0| \leq \lambda\}} \sum_{i=1}^n Y_i \mathbf{1}\{|x_i - x_0| \leq \lambda\}.$$

# points $x_i$ that are $\lambda$-close to $x_0$ $=$

$\{|x_i - x_0| \leq \lambda, \ \forall i\}$ $\Leftrightarrow$ $\left\{\frac{|x_i - x_0|}{\lambda} \leq 1, \ \forall i\right\}$

A weighted average! Choose other weights? Kernel estimator:

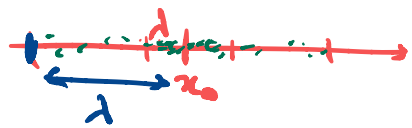
$$\hat{g}(x_0) = \frac{1}{\sum_{i=1}^n K\left(\frac{x_i - x_0}{\lambda}\right)} \sum_{i=1}^n Y_i K\left(\frac{x_i - x_0}{\lambda}\right) = \frac{1}{\sum_{i=1}^n w_i} \sum_{i=1}^n w_i Y_i.$$

use $K$ instead of $\mathbf{1}(\cdot)$

$\#$ points

$w_i = K\left(\frac{x_i - x_0}{\lambda}\right)$

- $K$ is a kernel function: specifies the nature of the local neighbourhood at each point $(x_0)$
  - ↪ E.g. standard Gaussian pdf, $K(x) = \varphi(x)$
- $\lambda$ is the bandwidth parameter: controls the width of the window
  - ↪ small $\lambda$ gives local behaviour, large $\lambda$ gives global behaviour
- Choice of $K$ not so important, choice of $\lambda$ very important.




- The resulting fitted values are linear in the responses, i.e., $\hat{\boldsymbol{Y}} = \boldsymbol{S}_\lambda \boldsymbol{Y}$, where the smoothing matrix $\boldsymbol{S}_\lambda$ depends on $x_1, \ldots, x_n$, $K$, and $\lambda$. Analogous to a projection matrix in linear regression, but $\boldsymbol{S}_\lambda$ is not a projection.

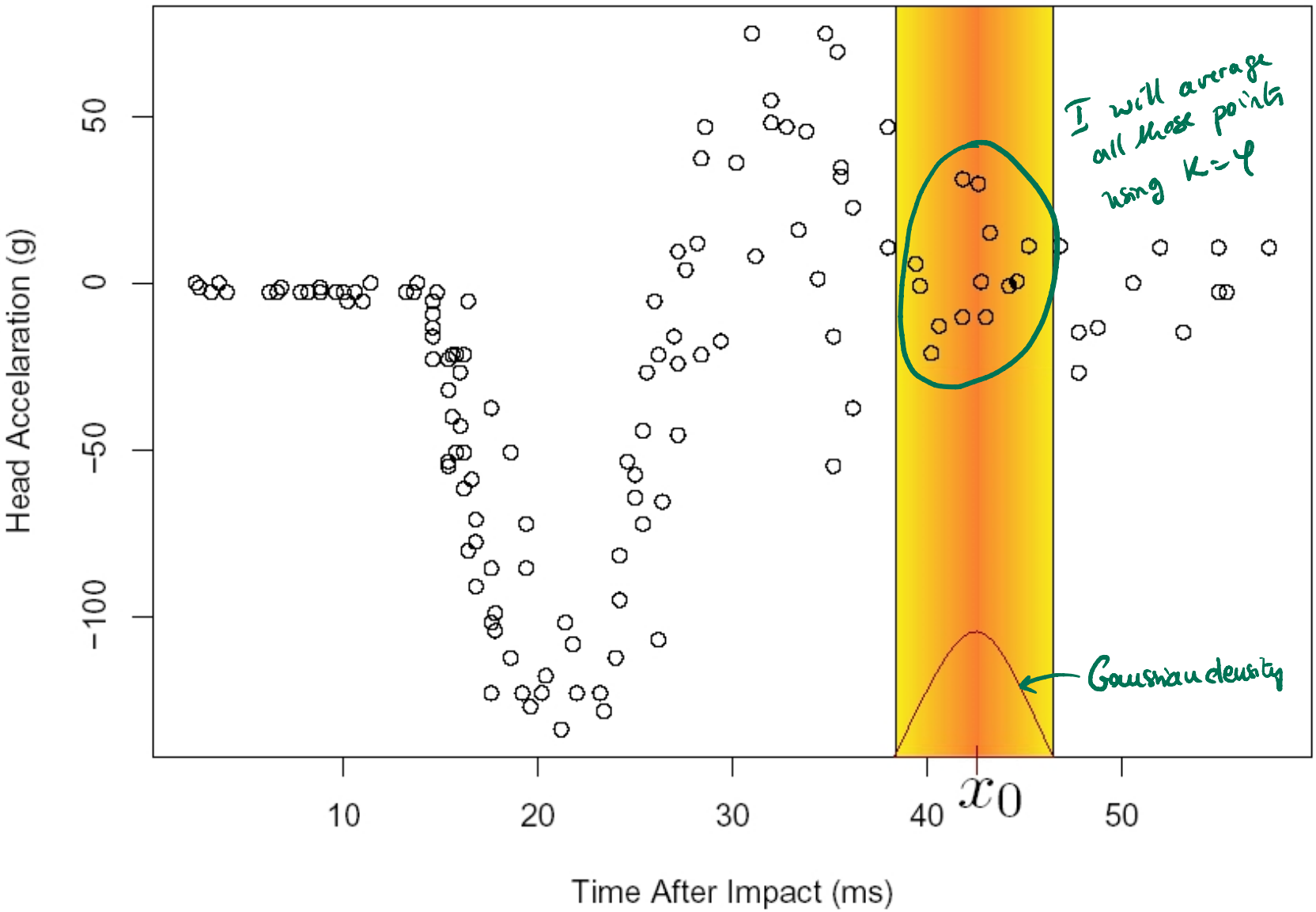



Figure: Visualising a kernel smoother at work

```
> plot(time,accel,xlab="Time After Impact (ms)",ylab="Head Accelaration (g)")
> lines(ksmooth(time,accel,kernel="normal",bandwidth=0.7))
> lines(ksmooth(time,accel,kernel="normal",bandwidth=5),col="red")
> lines(ksmooth(time,accel,kernel="normal",bandwidth=10),col="blue")
```

small

$\lambda$

large

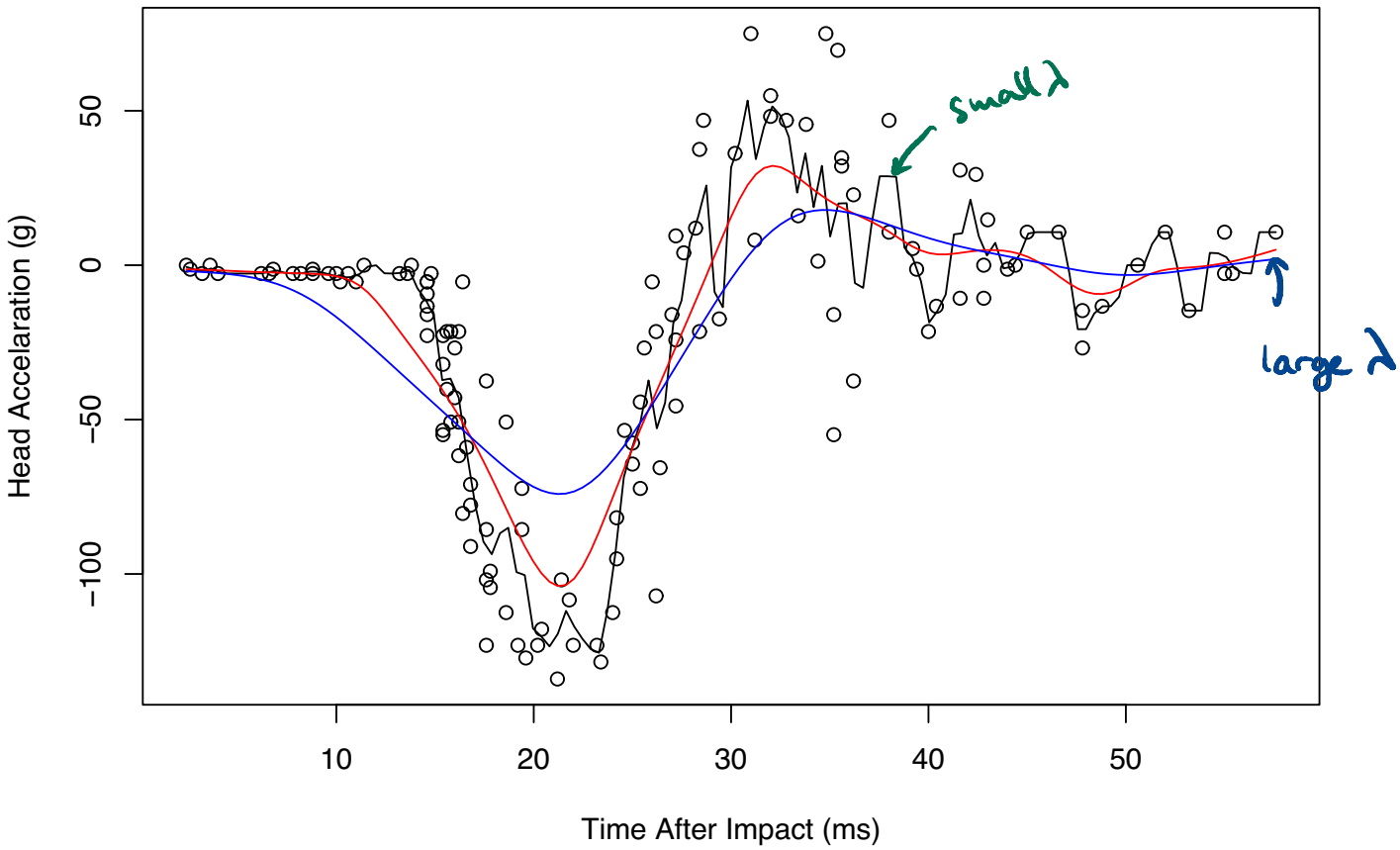



Figure: Motorcycle data kernel smooth for varying bandwidths

Whatever happened to likelihood, though? Find $h \in C^2$ that minimises

$$\underbrace{\sum_{i=1}^{n}\{Y_i - h(x_i)\}^2}_{\text{Fit Penalty}} + \underbrace{\lambda \int_I \{h''(t)\}^2 dt}_{\text{Roughness Penalty}}$$

penalization parameter

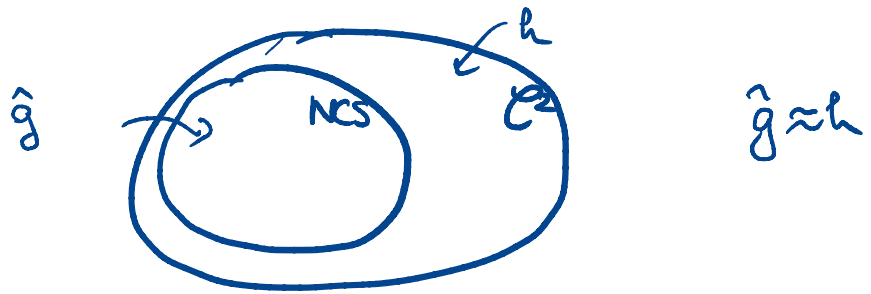
$h \in C^2$

$\lambda$: avoid large values of the penalty term as function of $h$

@ $h(t) = \alpha t + \beta$

$\Rightarrow h''(t) = 0$

- This is a Gaussian likelihood with a roughness penalty
  - ↪ If use only likelihood, any interpolating function is an MLE!
- $\lambda$ to balance fidelity to the data and smoothness of the estimated $h$ - why?
- What are the solutions when having $\lambda = 0$ or $\infty$?




$\hat{g} \approx h$

Remarkably, problem has unique explicit solution! I approximate $h$ by a function $g \in$ Nat. cubic splines.

$\hookrightarrow$ Natural Cubic Spline with knots at $\{x_i\}_{i=1}^n$:

- piecewise polynomials of degree 3,
- with pieces defined at the knots,
- with two continuous derivatives at the knots,
- and linear outside the data boundary.

Example: Suppose $n = 2$, then knots are at $x_1, x_2$, and the cubic spline is defined by the basis functions $1, X, X^2, X^3, X \mapsto (X - x_1)_+^3$, and $X \mapsto (X - x_2)_+^3$. ] functions are the basis functions

NB: There are $n$ knots - is it overparametrized? Think about the penalization.

$\hat{g}(x) = \sum_{k \le 6} \theta_k \xi_k(x)$ $\hat{g}$ estimates $h$ of previous slide.

Can represent splines via natural spline basis functions $B_j$, as

$$s(x) = \sum_{j=1}^{n} \gamma_j B_j(x).$$

Defining matrices $\boldsymbol{B}$ and $\boldsymbol{\Omega}$ as

$$B_{ij} = B_j(x_i), \quad \Omega_{ij} = \int B_i''(x) B_j''(x) dx,$$

our penalised likelihood becomes

$$\min \left\{ (\boldsymbol{Y} - \boldsymbol{B}\gamma)^\top (\boldsymbol{Y} - \boldsymbol{B}\gamma) + \lambda \gamma^\top \boldsymbol{\Omega} \gamma \right\}.$$

Differentiating and equating with zero yields

$$(\boldsymbol{B}^\top \boldsymbol{B} + \lambda \boldsymbol{\Omega}) \hat{\gamma} = \boldsymbol{B}^\top \boldsymbol{Y} \implies \hat{\gamma} = (\boldsymbol{B}^\top \boldsymbol{B} + \lambda \boldsymbol{\Omega})^{-1} \boldsymbol{B}^\top \boldsymbol{Y}.$$

- We recognize a solution of a generalized ridge regression
- The *smoothing matrix* is $\boldsymbol{S}_\lambda = \boldsymbol{B}(\boldsymbol{B}^\top \boldsymbol{B} + \lambda \boldsymbol{\Omega})^{-1} \boldsymbol{B}^\top$.
- The cubic spline fit is approximately a kernel smoother (keyword: equivalent kernel).

```
lines(smooth.spline(time,accel),col="red")
```

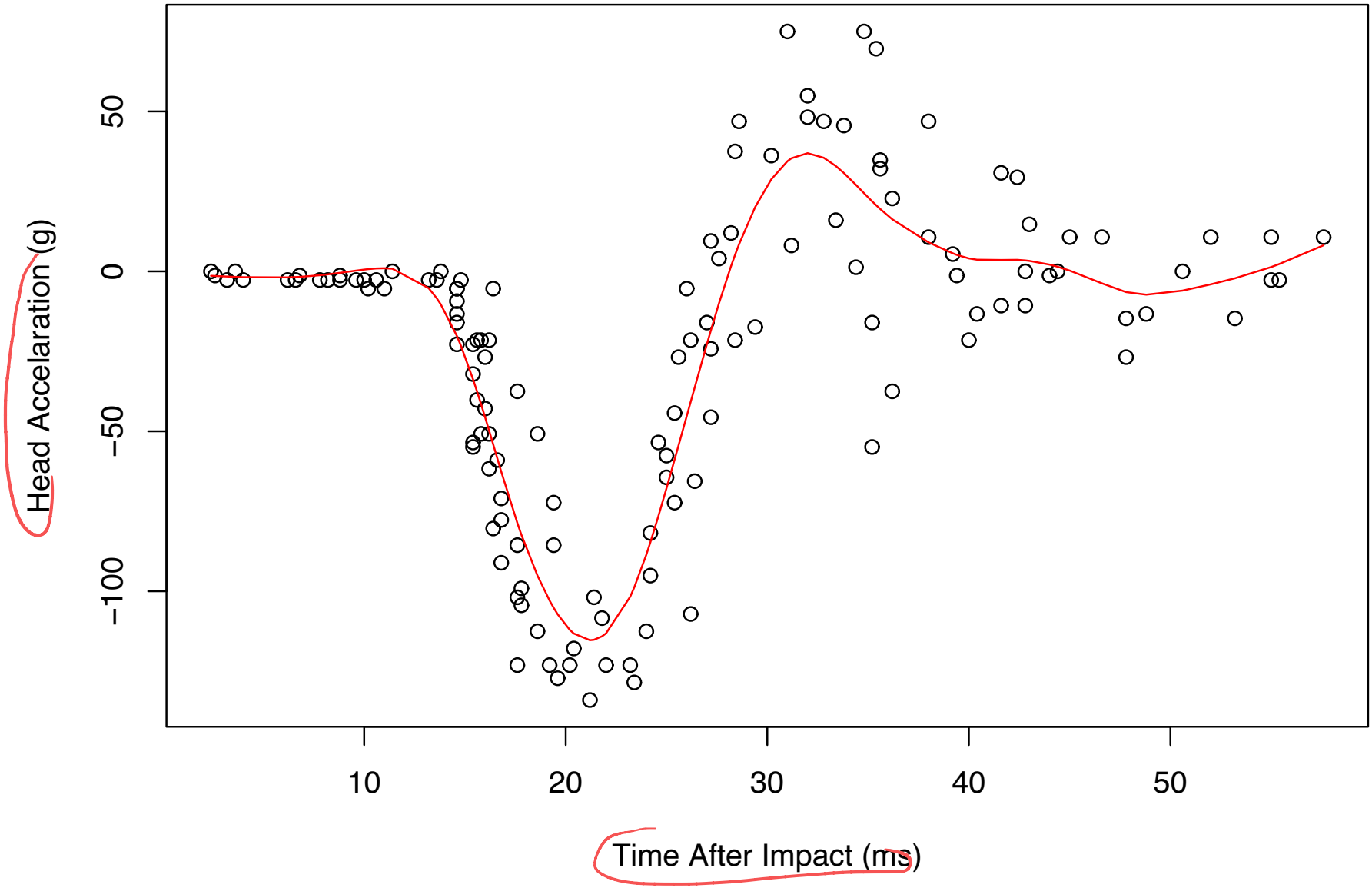



Figure: Motorcycle Example Cubic Spline Fit

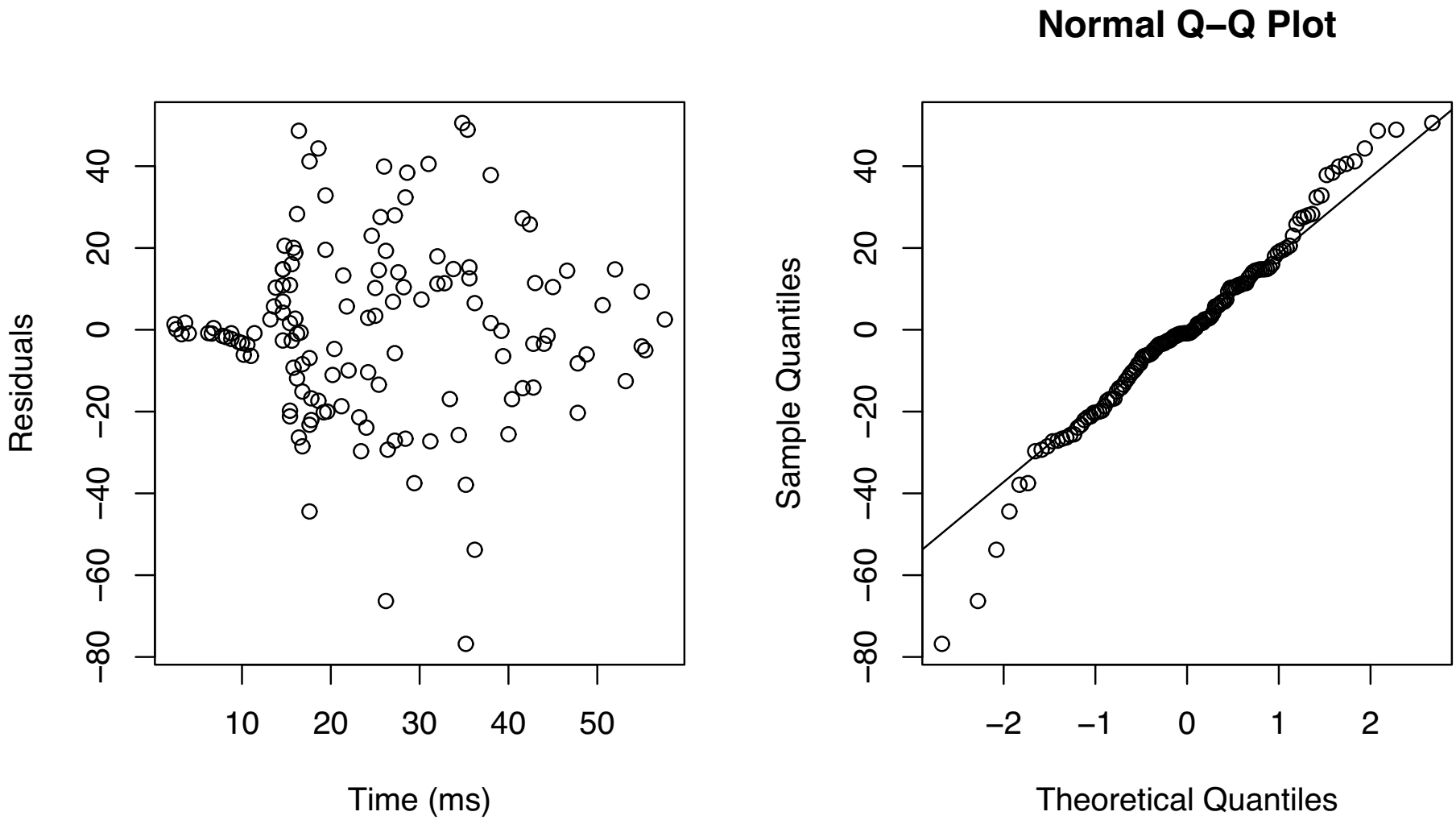



Figure: Motorcycle Example Cubic Spline Residuals

- Least squares estimation: $\boldsymbol{Y} = \boldsymbol{X}_{n\times p}\beta + \varepsilon$, we have $\hat{\boldsymbol{Y}} = \boldsymbol{H}\boldsymbol{Y}$, with $\mathrm{trace}(\boldsymbol{H}) = p$, in terms of the projection matrix $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top$.
- In spline smoothing

$$\hat{\boldsymbol{Y}} = \underbrace{\boldsymbol{B}(\boldsymbol{B}^\top\boldsymbol{B} + \lambda\Omega)^{-1}\boldsymbol{B}^\top}_{\boldsymbol{S}_\lambda}\ \boldsymbol{Y}.$$

  suggesting definition of equivalent degrees of freedom of smoother as

$$\mathrm{edf} = \mathrm{trace}(\boldsymbol{S}_\lambda)$$

- $\mathrm{trace}(\boldsymbol{S}_\lambda)$ is monotone decreasing in $\lambda$, with $\mathrm{trace}(\boldsymbol{S}_\lambda) \to 2$ as $\lambda \to \infty$ (will always have two nonzero eigenvalues) and $\mathrm{trace}(\boldsymbol{S}_\lambda) \to n$ as $\lambda \to 0$.
- Note 1–1 map $\lambda \leftrightarrow \mathrm{trace}(\boldsymbol{S}_\lambda) =$ df, so usually determine roughness using $\mathrm{edf}$ (interpretation easier).
- Each eigenvalue of $\boldsymbol{S}_\lambda$ lies in $(0, 1)$, so this is a smoothing matrix, not a projection matrix.

Focus on the fit for the given grid $x_1, \ldots, x_n$:

$$\hat{\mathbf{g}} = (\hat{g}(x_1), \ldots, \hat{g}(x_n)), \quad \mathbf{g} = (g(x_1), \ldots, g(x_n))$$

Consider the mean squared error:

$$\mathbb{E}(\|\mathbf{g} - \hat{\mathbf{g}}\|^2) = \underbrace{\mathbb{E}\{\|\mathbb{E}(\hat{\mathbf{g}}) - \hat{\mathbf{g}}\|^2\}}_{\text{variance}} + \underbrace{\|\mathbf{g} - \mathbb{E}(\hat{\mathbf{g}})\|^2}_{\text{bias}^2}.$$

$= \mathbb{E}\left[\|S_\lambda \mathbb{E}Y - S_\lambda Y\|^2\right] + \|g - S_\lambda \mathbb{E}Y\|^2$

In the case of a linear smoother, for which $\hat{\mathbf{g}} = \boldsymbol{S}_\lambda \boldsymbol{Y}$, we easily calculate

$$\mathbb{E}(\|\mathbf{g} - \hat{\mathbf{g}}\|^2) = \frac{\operatorname{trace}(\boldsymbol{S}_\lambda \boldsymbol{S}_\lambda^\top)}{n}\sigma^2 + \frac{(\mathbf{g} - \boldsymbol{S}_\lambda \mathbf{g})^\top (\mathbf{g} - \boldsymbol{S}_\lambda \mathbf{g})}{n},$$

$Y = g(x) + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2)$

so

- $\lambda \uparrow \implies$ variance $\downarrow$ but bias $\uparrow$,
- $\lambda \downarrow \implies$ bias $\downarrow$ but variance $\uparrow$.
- Would like to choose $\lambda$ to find optimal bias-variance tradeoff:
  - ↪ Unfortunately, optimal $\lambda$ will depend on unknown $\boldsymbol{g}$!

- Fitted values are $\hat{\boldsymbol{Y}} = \boldsymbol{S}_\lambda \boldsymbol{Y}$.

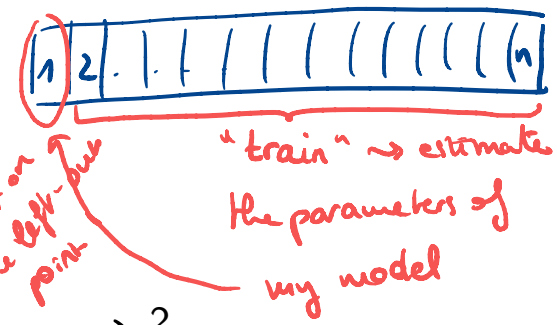
LR: $\hat{Y} = HY$

- Fitted value $\hat{Y}_j^-$ obtained when $(Y_j, x_j)$ is dropped from fit is

from training

$$S_{jj}(\lambda)(Y_j - \hat{Y}_j^-) = \hat{Y}_j - \hat{Y}_j^-.$$

$\{(Y_i, x_i)\}$ « $\mathcal{D}_n$




- Cross-validation sum of squares is

$$\mathrm{CV}(\lambda) = \sum_{j=1}^{n} (Y_j - \hat{Y}_j^-)^2 = \sum_{j=1}^{n} \left\{ \frac{Y_j - \hat{Y}_j}{1 - S_{jj}(\lambda)} \right\}^2,$$

and generalised cross-validation sum of squares is

$$\mathrm{GCV}(\lambda) = \sum_{j=1}^{n} \left\{ \frac{Y_j - \hat{Y}_j}{1 - \mathrm{trace}(S_\lambda)/n} \right\}^2,$$

where $S_{jj}(\lambda)$ is $(j,j)$ element of $\boldsymbol{S}_\lambda$.

If $\mathcal{F} \ni g(\cdot)$ is a separable Hilbert space, we can write:

$$g(x) = \sum_{k \in \mathbb{Z}} \beta_k \psi_k(x) \quad \text{(in an appropriate sense)},$$

$Y = g(x) + \varepsilon$

$\beta^T x \rightarrow$ used orthogonality onto the column space of $X$

with $\{\psi\}_{k=1}^{\infty}$ known (orthogonal) basis functions for $\mathcal{F}$, e.g.,

- $\mathcal{F} = L^2(-\pi, \pi)$,
- $\{\psi_k\} = \{e^{-ikx}\}_{k \in \mathbb{Z}}$, $\psi_i \perp \psi_j$, $i \neq j$.
- Gives Fourier series expansion, $\beta_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} g(x) e^{-ikx} dx$.

If we truncate series, then we reduce to linear regression:

$$Y_i = \overbrace{\sum_{|k| < \tau} \beta_k \psi_k(x_i)}^{\sum g(x_i)} + \varepsilon_i, \quad \tau < \infty$$

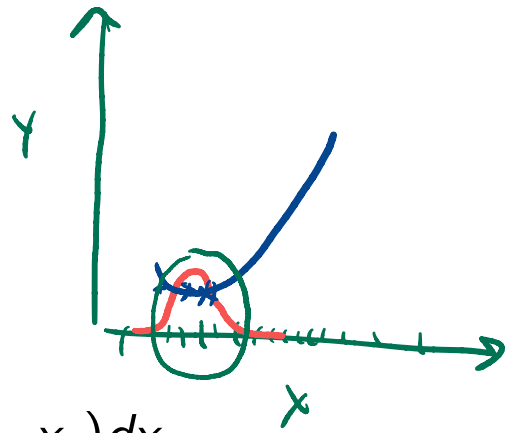
Notice: truncation has implications, e.g., in Fourier case:

- Truncating implies assume $g \in \text{span}\{\psi_{-\tau}, ..., \psi_{\tau}\} \subset L^2$.
- Interpret this as a smoothness assumption on $g$.
- How to choose $\tau$ optimally?

Classical exercise in Fourier analysis shows that

$$\sum_{k=-\tau}^{\tau} \beta_k e^{-ikx} = \frac{1}{2\pi} \int_{-\pi}^{\pi} g(y) D_\tau(x-y) dy$$

with the Dirichlet kernel of order $\tau$, $D_\tau(u) = \frac{\sin\{(\tau+1/2)u\}}{\sin(u/2)}$.




Recall kernel smoother:

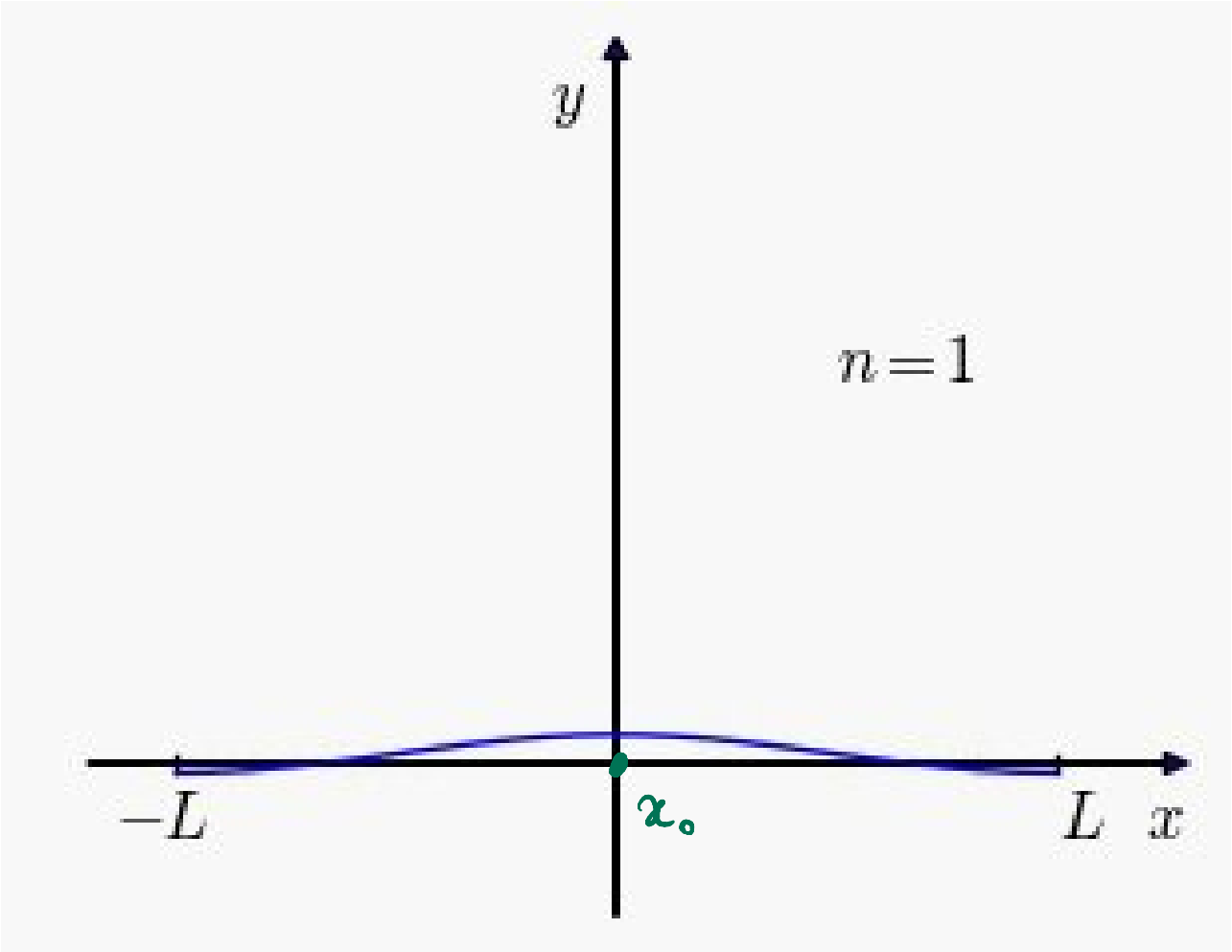
$$\hat{g}(x_0) = \sum_{i=1}^{n} \frac{Y_i K_\lambda(x_i - x_0)}{\sum_{i=1}^{n} K_\lambda(x_i - x_0)} = \frac{1}{c} \int_I y(x) K_\lambda(x - x_0) dx,$$

with

$$y(x) = \sum_{i=1}^{n} Y_i \delta(x - x_i).$$

- So if $K$ is the Dirichlet kernel, we can do series approximation via kernel smoothing.
- Works for other series expansions with other kernels (e.g., Fourier with convergence factors)

# The Dirichlet kernel



So far: how to estimate $g : \mathbb{R} \to \mathbb{R}$ (assumed smooth) in

$$Y_i = g(x_i) + \varepsilon_i, \quad \varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2), \text{ given data } \quad \{(Y_i, x_i)\}_{i=1}^n.$$

Can extend to GLM setting as:

$$Y_i | x_i \overset{indep}{\sim} \exp\{g(x_i)y - \gamma(g(x_i)) + S(y)\}$$

- Parametrise candidate $g$ via spline

$$s(x) = \sum_{j=1}^{n} \gamma_j B_j(x).$$

parameters to estimate $\in \mathbb{R}$

spline functions fixed and depend on the covariates $\equiv$ knots of the fct.

- Define matrices $\boldsymbol{B}$ and $\boldsymbol{\Omega}$ as before,

$$B_{ij} = B_j(x_i), \quad \Omega_{ij} = \int B_i''(x) B_j''(x) dx$$

- And consider penalised likelihood, similarly as with penalised GLM

$\hat{\gamma} \in \operatorname{argmin}_{\gamma \in \Theta \subset \mathbb{R}^n}$ $\ell_n(\gamma) + \lambda \gamma^\top \boldsymbol{\Omega} \gamma = \gamma^\top \boldsymbol{B}^\top \boldsymbol{Y} - \sum_{i=1}^{n} \gamma(\boldsymbol{b}_i^\top \gamma) + \lambda \gamma^\top \boldsymbol{\Omega} \gamma.$

How can we generalise to multivariate covariates?

- "Immediate" Generalisation: $g : \mathbb{R}^p \to \mathbb{R}$ (smooth)

$$Y_j = g(x_{j1}, \ldots, x_{jp}) + \varepsilon_j, \quad \varepsilon_j \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$$

- Estimation by (e.g.) multivariate kernel method.

$(X^\top X)$

- Two basic drawbacks of this approach ...

↪ Shape of kernel? (definition of *local*)

↪ *Curse of dimensionality*

What is "local" in $\mathbb{R}^p$, though?

$\rightarrow$ Need some definition of "local" in the space of covariates

$\hookrightarrow$ Use some metric on $\mathbb{R}^p \ni (x_1, \ldots, x_p)^\top$ !

But which one?

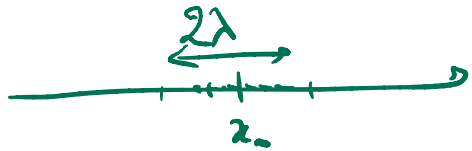
- Choice of metric $\iff$ choice of geometry
  - $\hookrightarrow$ e.g., curvature reflects intertwining of dimensions
- Geometry $\implies$ reflects structure in the covariates
  - potentially different units of measurement (variable stretching of space)
  - $g$ may be of higher variation in some dimensions (need finer neighbourhoods there)
  - statistical dependencies present in the covariates ("local" should reflect these)

## Curse of Dimensionality

*"neighbourhoods with a fixed number of points become less local as the dimensions increase"*

*Bellman (1961)*

- Notion of local in terms of % of data: fails in high dimensions
  $\hookrightarrow$ There is too much space!
- Hence to allow for reasonably small bandwidths
  $\hookrightarrow$ Density of sampling must increase.
- Need to have ever larger samples as dimension grows.



Attempt to find a link/compromise between:

- our mastery of 1D case (at least we can do that well . . . ),
- and higher dimensional covariates (and associated difficulties).

Perhaps something that can be fitted/interpreted variable-by-variable?

- Compromise: Additive Model

$$Y_i = \alpha_i + \sum_{k=1}^{p} f_k(x_{ik}) + \varepsilon_i, \quad \varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2),$$

with $f_k$'s univariate smooth functions, $\sum_i f_k(x_{ik}) = 0$.

- Can extend to Generalised Additive Model:

$$Y_i | \boldsymbol{x}_i^\top \overset{indep}{\sim} \exp\left\{ \alpha_i y + y \sum_{k=1}^{p} f_k(x_{ik}) - \gamma\left( \alpha_i + \sum_{k=1}^{p} f_k(x_{ik}) \right) + S(y) \right\}$$

- How to fit additive model? Consider Gaussian case only for simplicity.
  - ↪ Know how to fit each $f_k$ separately quite well
  - ↪ Take advantage of this ...

- Consider $i$th response:

$$\mathbb{E}\left[Y_i - \alpha - \sum_{m \neq k} f_m(x_{im})\right] = f_k(x_{ik})$$

- Suggests the *Backfitting Algorithm*:

(1) `Initialise:` $\alpha = \bar{Y}$, $f_k = f_k^0$, $k = 1, \ldots, p$.

(2) `Cycle:` Get $f_k$ by 1D smoothing of partial residual scatterplot

$$\left\{\left(Y_i - \alpha - \sum_{m \neq k} f_m(x_{im}), x_{ik}\right)\right\}_{i=1}^{n} = \{e_{ik}, x_{ik}\}_{i=1}^{n}.$$

(3) `Stop:` when individual functions don't change

- Any smoother can be used, usually splines.

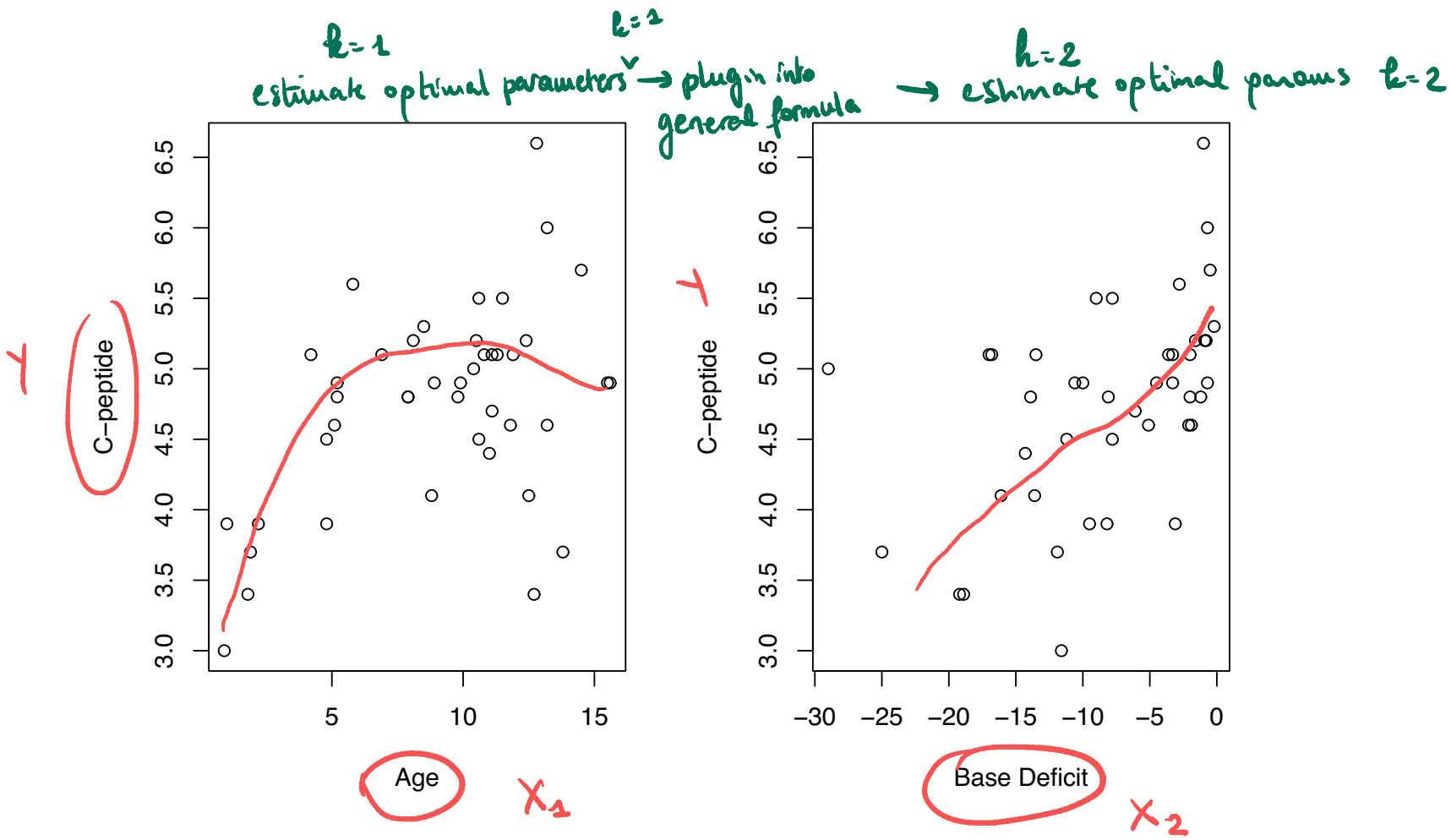



Figure: Example: Diabetes Data

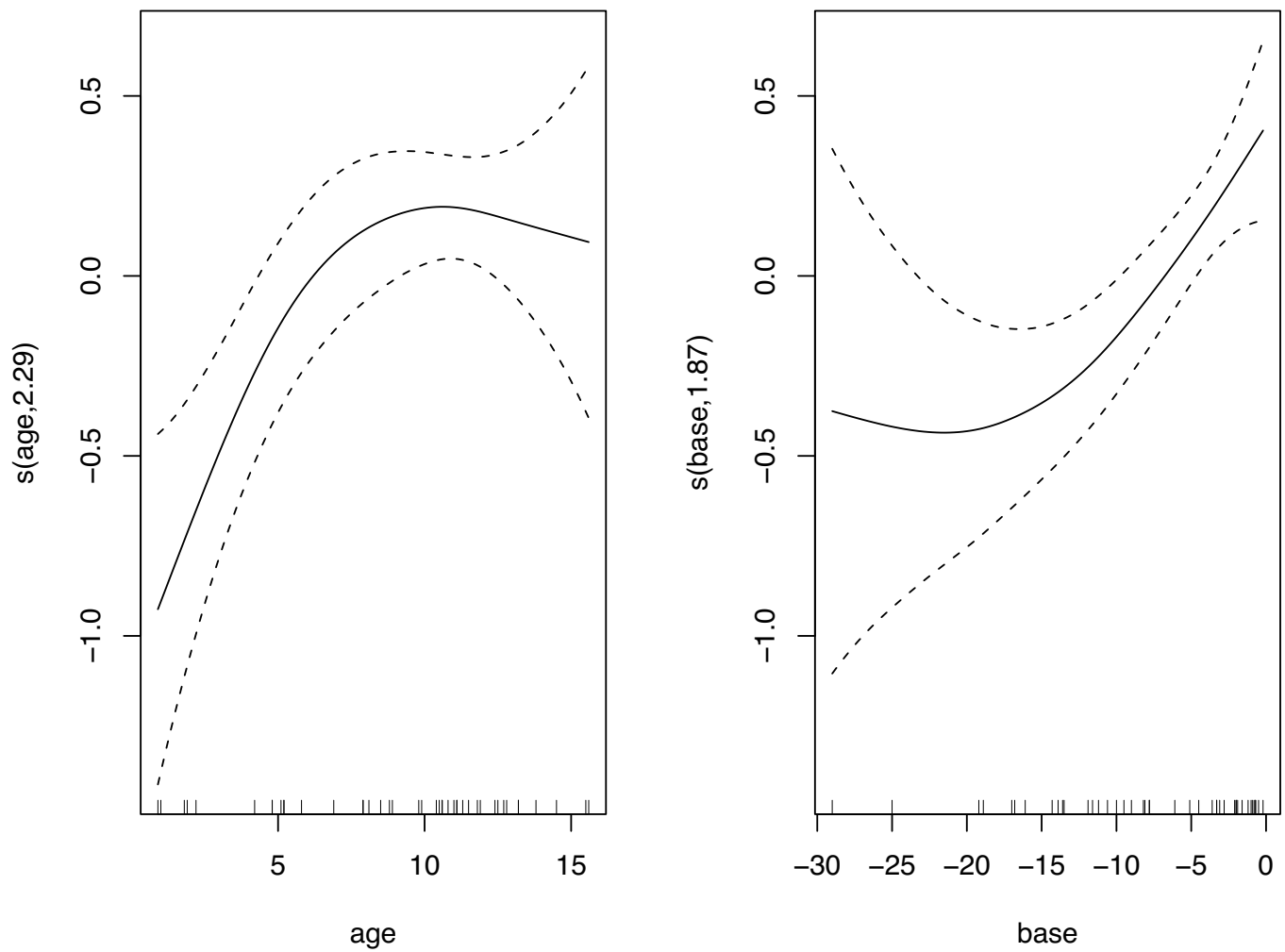



Figure: Example: Diabetes Data Additive Fit

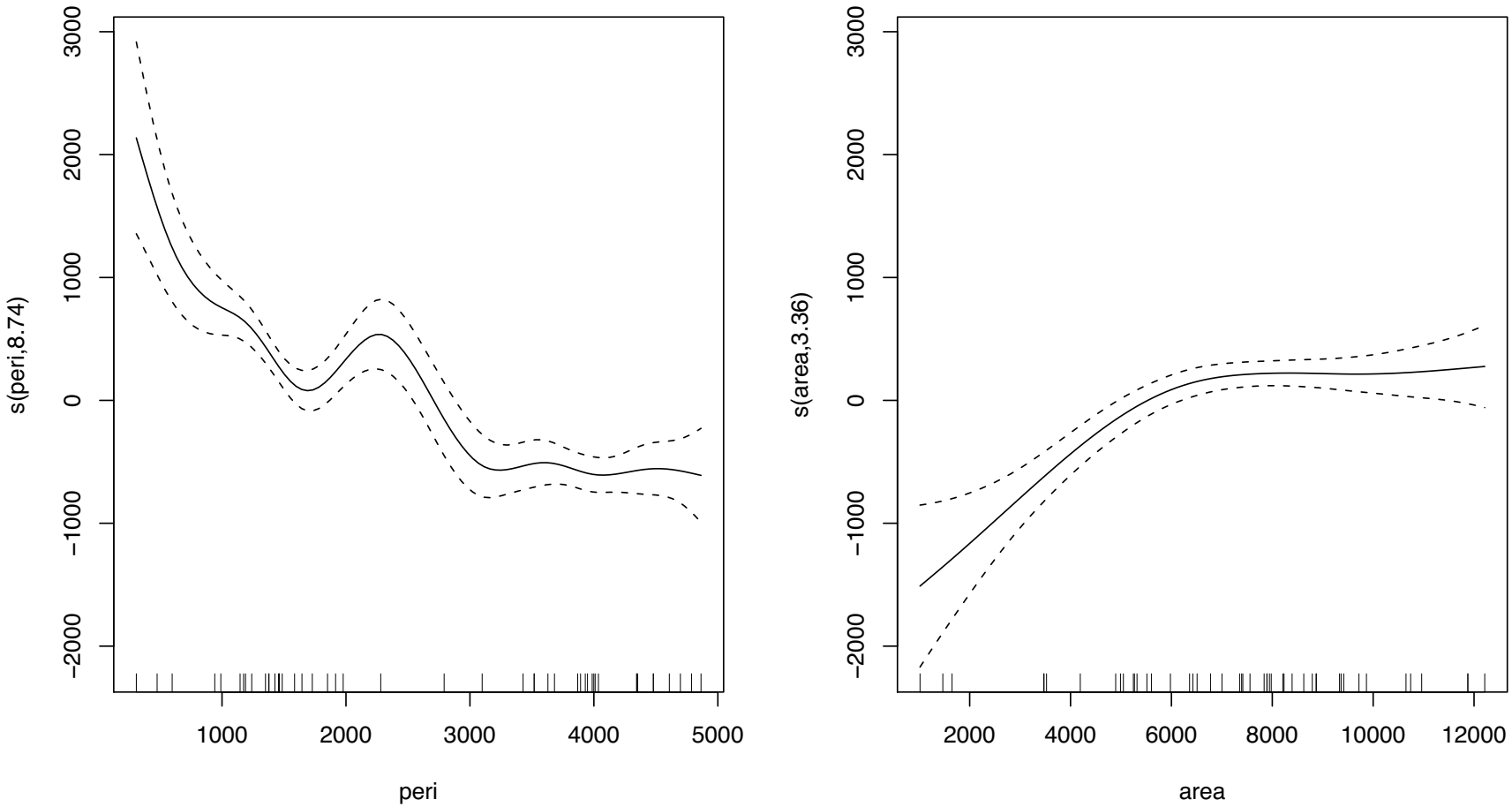


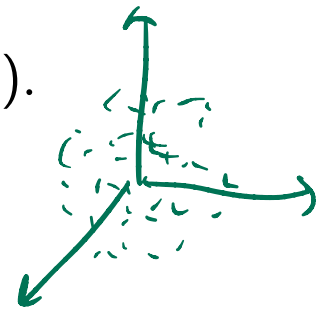

Figure: Example: Rock Permeability Data (measurements on 48 rock samples from a petroleum reservoir)

```
Family: gaussian
Link function: identity

Formula:
perm ~ 1 + s(peri) + s(area)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   415.45      27.18   15.29   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
          edf Est.rank      F  p-value
s(peri) 8.739        9 18.286 9.49e-11 ***
s(area) 3.357        7  6.364 7.41e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.815   Deviance explained = 86.3%
```

A different approach is inspired by tomography. Model Gaussian response as:

$$Y_i = \underbrace{\sum_{k=1}^{K} h_k(\boldsymbol{x}_i^\top \boldsymbol{\beta}_k)}_{=g(\boldsymbol{x}_i^\top)} + \varepsilon_i, \quad \|\boldsymbol{\beta}_k\| = 1, \ \varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2).$$

$\mathbb{R}^3$ : $\beta_1 = \begin{pmatrix}1\\0\\0\end{pmatrix}$ $\begin{pmatrix}0\\1\\0\end{pmatrix}$ $\begin{pmatrix}0\\0\\1\end{pmatrix}$



- Also additively decomposes $g$ into smooth functions $h_k : \mathbb{R} \to \mathbb{R}$.
- But each function now depends on a global linear feature $\boldsymbol{x}_i^\top \boldsymbol{\beta}_k$
  $\hookrightarrow$ a linear combination of the covariates
  $\hookrightarrow$ $\|\boldsymbol{\beta}_k\| = 1$ for identifiability.
- Projections directions to be chosen for best fit (nonlinear problem)
- Each $h_k$ is a ridge function of $\boldsymbol{x}_i^\top$: varies only in the direction defined by $\boldsymbol{\beta}_k$

Pros and Cons:

(+) By classical Fourier series, can show that any $C^1([0,1]^p) \to \mathbb{R}$ function is uniformly approximated arbitrarily well as $K \to \infty$. Useful for prediction.

(−) Interpretability? What do terms mean within problem?

## How is the model fitted to data?

Assume only one term, $K = 1$ and consider penalized likelihood:

$$\min_{h_1 \in C^2[0,1], \|\boldsymbol{\beta}\|=1} \left\{ \sum_{i=1}^{n} \{Y_i - h_1(\boldsymbol{x}_i^\top \boldsymbol{\beta})\}^2 + \int_0^1 \{h_1''(t)\}^2 dt \right\}.$$

Two steps:

- *Smooth*: Given a direction $\boldsymbol{\beta}$, fitting $h_1(\boldsymbol{x}_i^\top \boldsymbol{\beta})$ is done via 1D smoothing.
- *Pursue*: Given $h_1$, have a non-linear regression problem w.r.t. $\boldsymbol{\beta}$.

Hence, iterate between the two steps

↪ Complication is that $h_1$ not explicitly known, so need numerical derivatives.

↪ Computationally intensive (impractical in the '80's but doable today).

↪ Can separate second step by looking for non-Gaussian projection directions.

Further terms added in forward stepwise manner.

If $\boldsymbol{\beta}_k$ needs to be estimated non-linearly anyway...

$$g(\boldsymbol{x}_i^\top) \approx \sum_{k=1}^{K} h_k(\boldsymbol{x}_i^\top \boldsymbol{\beta}_k)$$

... do we really need to estimate the $h_k$ or can we fix them?

## Theorem (Nonlinear Sigmoidal Approximation)

*Let* $\Psi : \mathbb{R} \to [0,1]$ *be a strictly increasing distribution function and* $g : [0,1]^p \to \mathbb{R}$ *be an arbitrary continuous function. Then, for any* $\epsilon > 0$*, there exists* $K < \infty$ *and vectors* $\boldsymbol{\alpha}, \boldsymbol{t} \in \mathbb{R}^K$ *and* $\{\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K\} \subset \mathbb{R}^p$ *such that*

$$\sup_{\boldsymbol{x} \in [0,1]^d} \left| g(\boldsymbol{x}) - \sum_{k=1}^{K} \alpha_k \Psi(t_k + \boldsymbol{x}^\top \boldsymbol{\beta}_k) \right| < \epsilon.$$

- Can take $h_k$ to be translations of the same known function $\Psi$!
- The tradeoff is that $K$ may need to be quite large (interpretability?)
- Called a (single layer) neural network by analogy to synaptic function.
- A parametric model with many parameters – fit by nonlinear least squares (gradient descent)

What about including transformations of the original covariates?

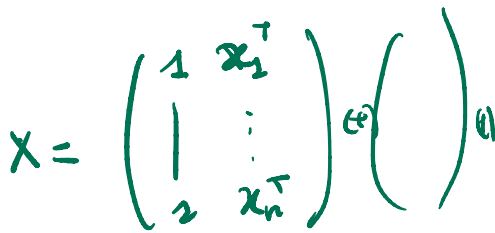
1. Can of course include $J$ transformations $w_j : \mathbb{R}^p \to \mathbb{R}$

$$(u_1, ..., u_p) \mapsto w_j(u_1, ..., u_n), \qquad j = 1, ..., J,$$

   of the original variables as additional covariates by suitably enlarging the design matrix $\boldsymbol{X}$.

2. We simply adjoin to $\boldsymbol{X}$ another $J$ columns of dimension $n \times 1$ each:

$$\begin{pmatrix} w_j(\boldsymbol{x}_1^\top) \\ \vdots \\ w_j(\boldsymbol{x}_n^\top) \end{pmatrix} \qquad j = 1, ..., J.$$



3. Which functions $w_j$ should we pick though?

Since we've gone nonlinear anyway,

why not attempt to learn which transformations to include from the data?

How?

- Instead of including our original covariates ($p$ columns of $\boldsymbol{X}$)...
- ... use $q$ derived covariates ($q$ can be larger than $p$)

$$\begin{pmatrix} w_1(\boldsymbol{x}_1^\top) \\ \vdots \\ w_1(\boldsymbol{x}_n^\top) \end{pmatrix}, \begin{pmatrix} w_2(\boldsymbol{x}_1^\top) \\ \vdots \\ w_2(\boldsymbol{x}_n^\top) \end{pmatrix}, \quad \cdots \quad , \begin{pmatrix} w_q(\boldsymbol{x}_1^\top) \\ \vdots \\ w_q(\boldsymbol{x}_n^\top) \end{pmatrix}$$

- ... where the $q$ transformations $\{w_j\}_{j=1}^q$ are to be estimated from the data.

Recycling our nonlinear approximation theorem, write

$$w_j(\boldsymbol{x}^\top) \approx \sum_{m=1}^{M_j} \delta_{m,j} \Psi(s_{m,j} + \boldsymbol{x}^\top \gamma_{m,j})$$

using the same $\Psi$, and needing to estimate $(\boldsymbol{\delta}_j, \boldsymbol{s}_j, \gamma_{1,j}, ..., \gamma_{M_j,j})$, for $j = 1, ..., q$.

Assuming that we've constructed our new variables, we have a new design matrix

$$\begin{pmatrix} w_1(\boldsymbol{x}_1^\top) & \dots & w_q(\boldsymbol{x}_1^\top) \\ \vdots & & \vdots \\ w_1(\boldsymbol{x}_n^\top) & \dots & w_q(\boldsymbol{x}_n^\top) \end{pmatrix}.$$

Summarising, we have defined a hierarchical nonlinear regression model:

$$Y_i = \sum_{k=1}^{K} \alpha_k \Psi\Big(t_k + (w_i(\boldsymbol{x}_1^\top), ..., w_i(\boldsymbol{x}_n^\top))\boldsymbol{\beta}_k\Big) + \varepsilon_i =$$

$$= \sum_{k=1}^{K} \alpha_k \Psi\left(t_k + \left(\sum_{m=1}^{M_1} \delta_{m,1}\Psi(s_{m,1} + \boldsymbol{x}^\top\boldsymbol{\gamma}_{m,1}), ..., \sum_{l=1}^{M_q} \delta_{l,q}\Psi(s_{l,q} + \boldsymbol{x}^\top\boldsymbol{\gamma}_{l,q})\right)\boldsymbol{\beta}_k\right) + \varepsilon_i$$

... known these days as a two-layer neural network.

- Can add more layers ("deep neural network").
- Highly non-linear and non-convex – cascade of simple nonlinearities applied to linear transformations.
- More easily perceived visually through a graphical representation

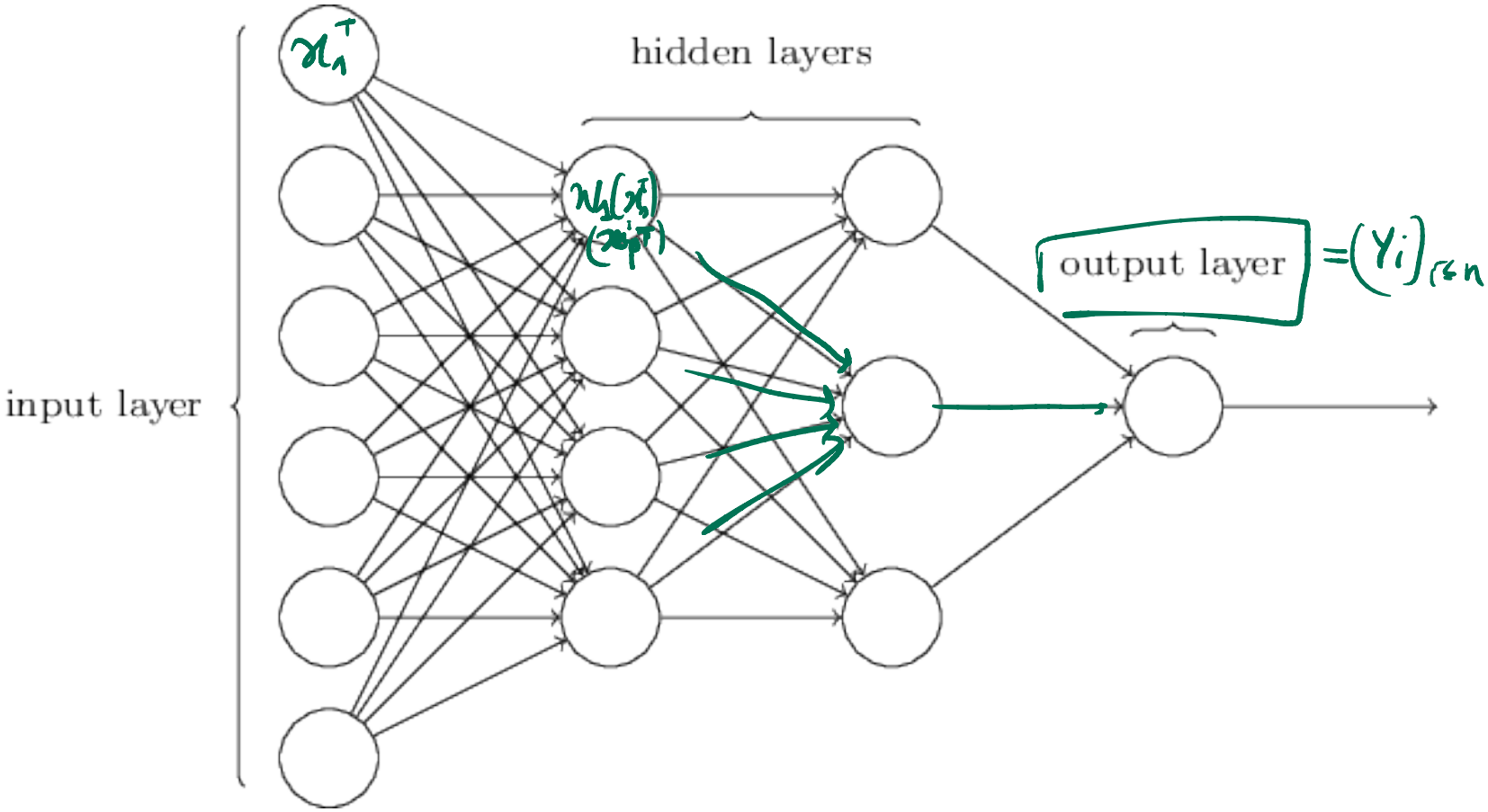



Figure: A multilayer neural network. The "input layer" corresponds to the original covariates. Each "hidden layer" corresponds to successively derived covariates via sigmoid superposition. The "output layer" is the final sigmoid superposition yielding the response.

Seems like we're asking a lot from the data...

...and indeed we are

- NN fitted by optimising least square fit of $Y_i$ to model w.r.t. parameters.
- Typically carried out with some flavour of gradient descent.
- Representation can be highly non-unique and many local optima can exist.
- Large number of parameters requires very large number of observations.
- Sampling distribution essentially totally unknown, practically no theory available.
- Seldom useful for interpretation – typically used for prediction/classification.
- Often requires context-dependent tuning and optimisation architecture.