STA314, Lecture 7

November 1, 2019

Stanislav Volgushev

The high-dimensional setting

The high-dimensional setting

In Statistics, we talk about the 'high-dimensional' setting when the number of predictors p is of comparable order or even larger than the sample size n (sometimes the latter is called ultra high dimensional).

Examples:

1. Genetics: predict risk of certain type of cancer from DNA mutations. Usually number of objects n is in the hundreds, but roughly $p = 500000$ common DNA mutations.
2. Marketing: for each customer in an online shop, have 0-1 encoding of items customer decided to buy or not to buy. Predict future shopping behaviour based on past purchase.

Clearly k-nn won't work in this setting. Which of the methods for linear regression discussed so far could be applicable?

'Usual' linear regression when $p > n$

Usual linear regression is not applicable when $p > n$ because there will be no unique solution to OLS problem. Recall: there is a unique minimizer of

$$RSS(b) = \sum_{i=1}^{n} (y_i - X_i^\top b)^2$$

if and only if matrix $\mathbf{X}^\top \mathbf{X}$ is invertible. Matrix $\mathbf{X}^\top \mathbf{X}$ is ...-dimensional and has rank at most .... If $p > n$ it can not be invertible.

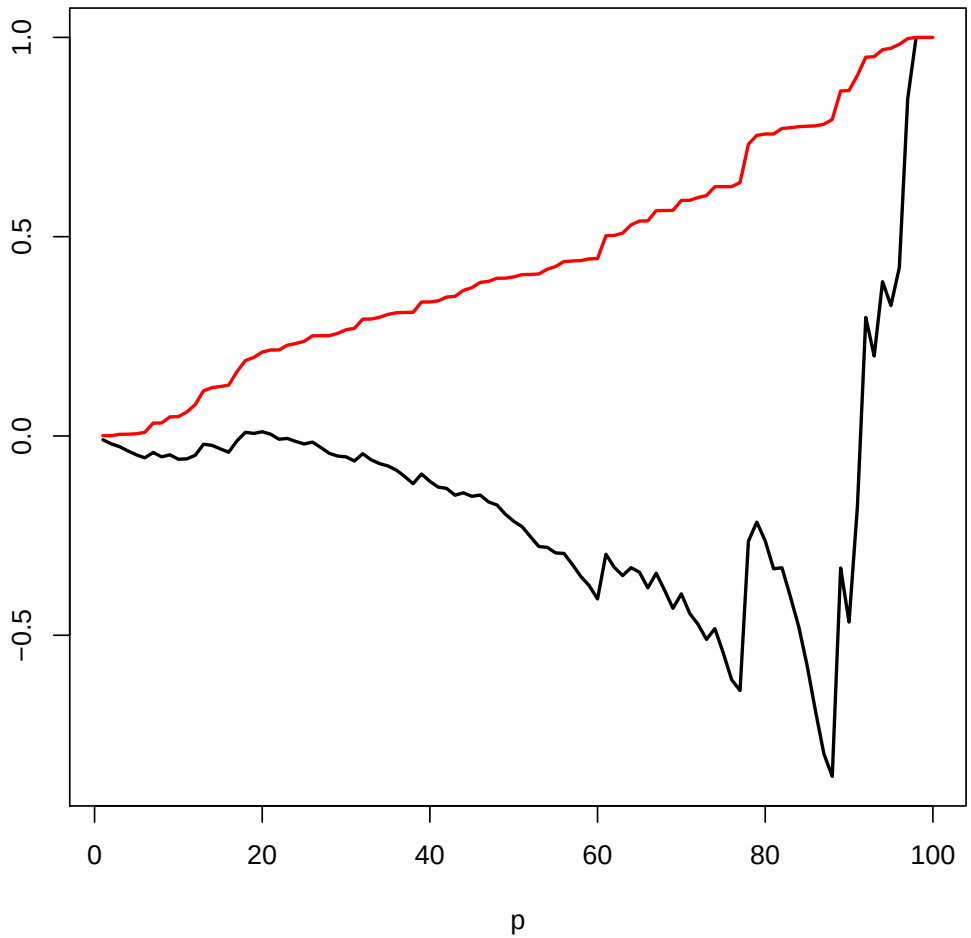
Even if p slightly smaller than n, linear regression will give a very small training error even if none of the predictors are relevant for response. Resulting test error will be very large.

Since usual least squares is not applicable, we need to look for *less flexible* models.

Examples: see discussion in lectures.

Adjusted R^2 fails for high dimensions

Setting: generate p predictors not related to the response, run linear regression. Sample size: $n = 100$. Plot below: R^2 and adjusted R^2 as a function of p. Which one is which? *R Exercise: generate this type of plot.*



Classical model selection methods

Backward stepwise selection: need to start with model that has all predictors, does not make sense when $p > n$.

Forward stepwise selection: can be used to select models as long as the largest model has less predictors than observations. So forward stepwise selection can be applied, but we need to stop before too many predictors selected.

Best subset: not feasible from a computational point of view. If there are p predictors, there are p choose k models with k predictors, that grows in p like p^k (can be very large!).

How do we compare models with different numbers of predictors?

- adjusted R^2 does not work for p close to n.
- Trouble with C_p, AIC, BIC: need $\hat{\sigma}^2$, difficult to obtain if $p > n$. The usual way to get this from 'full model' does not work, so not applicable when $p > n$.
- Cross-validation can be applied, but can also be expensive computationally.

Lasso, Ridge, PCR, PLS

Lasso: can often be applied even if $p > n$, provided λ is selected by cross-validation. In fact, one of the reasons for the great success of Lasso is the increased importance of data sets with $p > n$.

Ridge: will always lead to a unique solution if $\lambda > 0$, so can be applied when $p > n$. Disadvantage compared to Lasso: does set any coefficients to zero, so can not be used to perform model selection.

PCR, PLS: can be applied if $p > n$ but $M < n$ components are used.

Moving away from linearity

Moving away from linearity

In the last couple of lectures, we came up with different ways to build models that are *less flexible* compared to linear models in order to reduce the variance.

The main motivation was trading bias for variance.

from now on: models that are *more flexible* than just usual linear models. Motivation: try to model non-linear influence of predictor on outcome (for example through non-linear transformations of predictors) in order to decrease bias.

- *Basis expansions* (one predictor): popular classes of non-linear transformations of original predictor.
- *Smoothing splines*: similar idea to Basis expansion, but without some of the drawbacks.
- *Local regression*: generalize idea behind k-nn.
- *Generalized additive models (GAM)*: combining ideas mentioned above with several predictors.

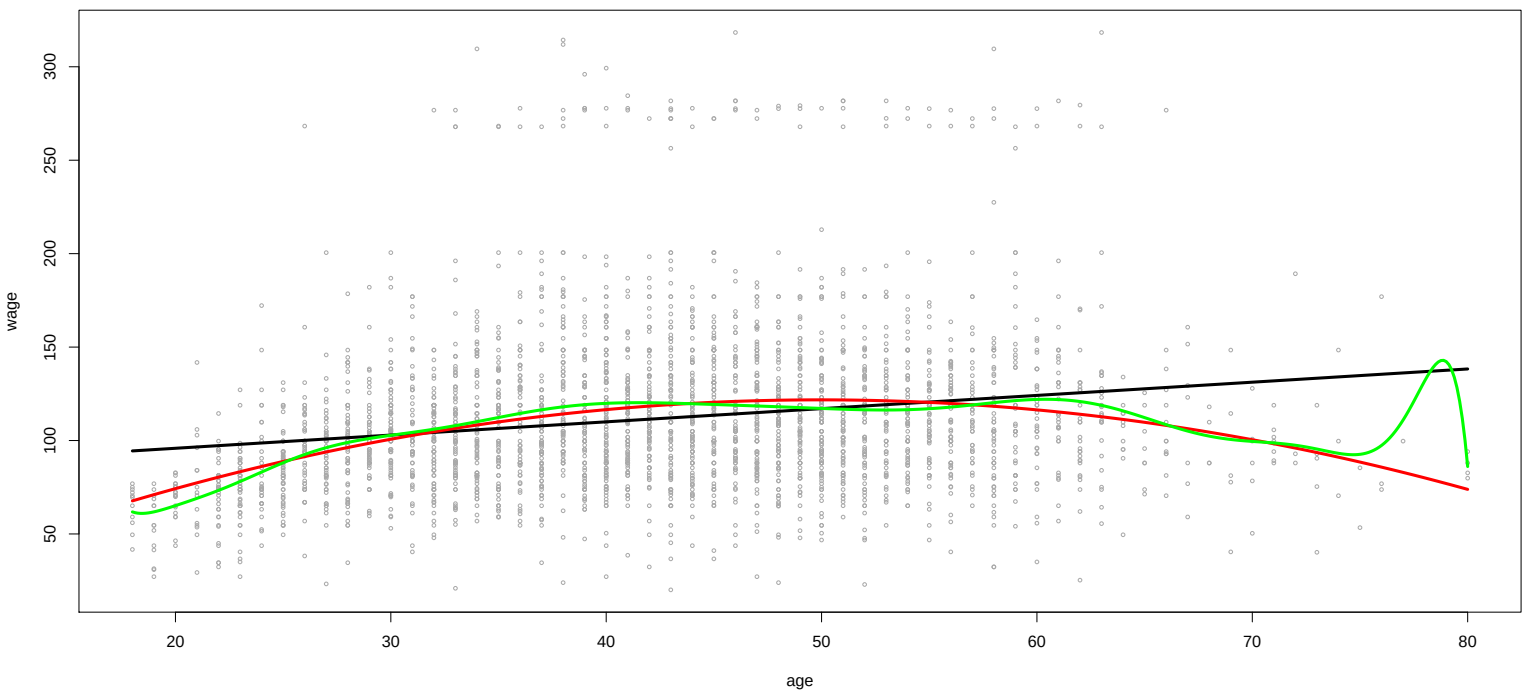
Non-linear transformations: polynomials

Have already seen polynomials earlier. Polynomial of degree d takes the form

$$f(x) = b_1 + b_2 x + b_3 x^2 + ... + b_d x^d.$$

Model can be learned using a linear model with new predictors, $x, x^2, ..., x^d$.

+ Simple to fit, can model non-linear relationships.
– Put a 'global' structure on the function f. Can be very wild near boundary points. Example below: degree 1 (black), 2 (red) and 15 (green).



Non-linear transformations: step functions

1. Divide range of predictor x into consecutive intervals $(c_1, c_2], (c_2, c_3], ..., (c_{K-1}, c_K]$ with each interval containing some values from the sample $x_1, ..., x_n$.
2. Define

$$\hat{f}(x) := \sum_{k=1}^{K-1} \frac{\sum_{i=1}^{n} y_i I\{x_i \in (c_k, c_{k+1}]\}}{\sum_{i=1}^{n} I\{x_i \in (c_k, c_{k+1}]\}} I\{x \in (c_k, c_{k+1}]\},$$

   here and in what follows: $I\{x_i \in (c_k, c_{k+1}]\} = 1$ if $x_i \in (c_k, c_{k+1}]$ and 0 else.

- This is the same as linear regression with new predictors $Z_{i,k} = I\{x_i \in (c_k, c_{k+1}]\}$, $k = 1, ..., K-1$ **without intercept** see lectures for details.
- In words: predicted value for $x \in (c_j, c_{j+1}]$ is mean of all y_i with $x_i \in (c_j, c_{j+1}]$.

Note: we need to select values of $c_1, ..., c_K$ (number and location). This can be done by 'eye-balling' the data (not very automatic...) or by putting values at quantiles of predictors and deciding about number by doing cross-validation.

\+ Simple to fit and interpret.

– Have discontinuities at breakpoints.

– Need to select breakpoints.

Non-linear transformations: piecewise polynomial regression

1. Divide range of predictor x into consecutive intervals $(c_1, c_2], (c_1, c_2], ..., (c_{K-1}, c_K]$ with each interval containing some values from the sample $x_1, ..., x_n$.
2. For $k = 1, ..., K-1$, let

$$\hat{B}_k := \arg \min_{B=(b_1,\ldots,b_{d+1})\in\mathbb{R}^{d+1}} \sum_{i:x_i\in(c_k,c_{k+1}]} (y_i - b_1 - b_2 x_i - ... - b_{d+1} x_i^d)^2$$

3. The predicted values are

$$\hat{f}(x) := \sum_{k=1}^{K-1} I\{x \in (c_k, c_{k+1}]\}(1, x, ..., x^d)\hat{B}_k$$

Graphical explanation: see lectures.

\+ Simple to fit and interpret.

\+ More flexible compared to step functions since shape between breakpoints is less constrained. Can require less breakpoints compared to step functions.

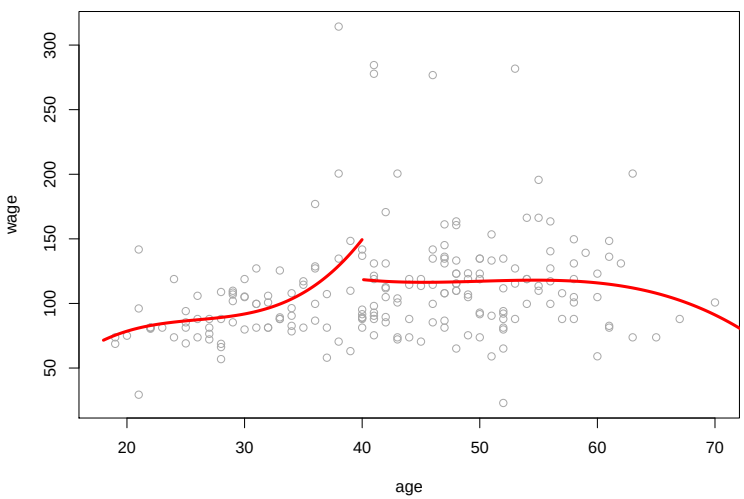
– Have jumps at breakpoints, need to manually select breakpoints.

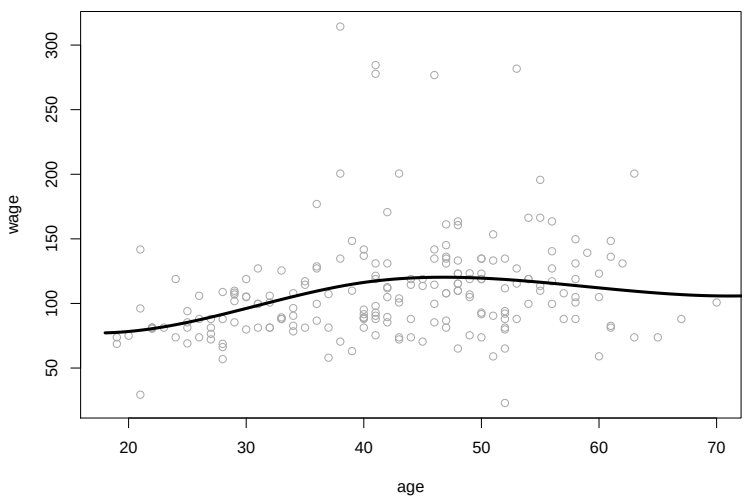
Polynomial splines and natural polynomial splines

One of the dis-advantages of fitting piecewise polynomials is that there are discontinuities at breakpoints. Can be problematic for interpretation. Potential solution: put constraints on polynomials that force them to connect in a continuous way, or even force derivatives to connect continuously.

Definition A *polynomial spline of degree d* with *knots* $c_1 < c_2 < ... < c_K$ is a function $g : \mathbb{R} \to \mathbb{R}$ with the following properties

1. On each interval $(-\infty, c_1), [c_1, c_2], ..., [c_K, \infty)$ g is a polynomial of degree $\leq d$ (the degree of a polynomial is the highest power of x with non-zero coefficient).
2. The function g is $d - 1$ times continuously differentiable on $\mathbb{R}$ (0 times continuously differentiable interpreted as continuous).





Cubic splines

Popular approach: cubic splines and natural cubic splines.

- *Cubic splines* are splines of degree 3.
- *Natural cubic splines* have the additional constraint that g is linear on $(\infty, c_1]$ and $[c_K, \infty)$.

Graphical explanation: see blackboard.

Theorem Any cubic spline g with knots $c_0, ..., c_K$ can be represented as

$$g(x) = \beta_1 + \beta_2 x + \beta_3 x^2 + \beta_4 x^3 + \beta_5 (x - c_1)_+^3 + ... + \beta_{K+4} (x - c_K)_+^3$$

where

$$(x - c_1)_+^3 := \begin{cases} x - c_1, & x - c_1 > 0 \\ 0, & x - c_1 \leq 0. \end{cases}$$

- Interpretation: fitting cubic splines with given knots can be achieved by fitting a linear model with transformed predictor x.
- Similar result is true for natural cubic splines and for polynomial splines of arbitrary degree.

Degrees of freedom for splines

The *degrees of freedom* of a spline correspond to the number of parameters in a linear model that describes the spline. It depends on the number of knots, the degree of the spline, and on whether the spline is a natural spine. Examples (answers in lectures):

1. A linear spline with 1 knot.

2. A degree d spline with 1 knot.

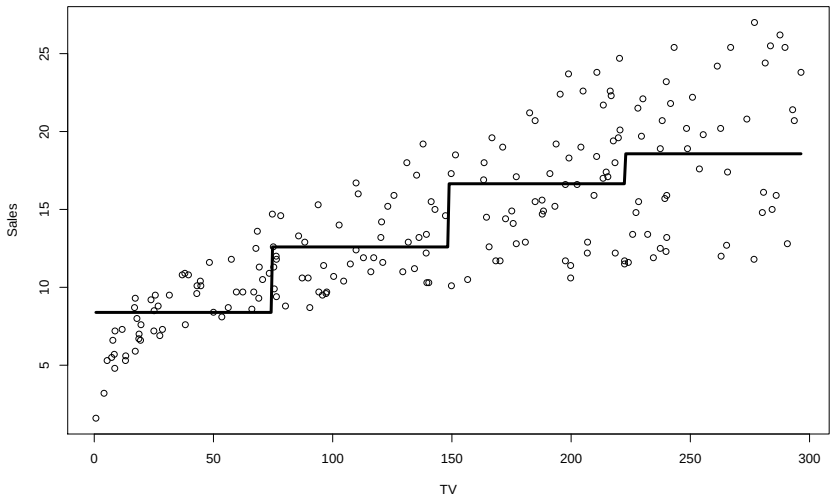
3. A cubic spline with 3 knots.

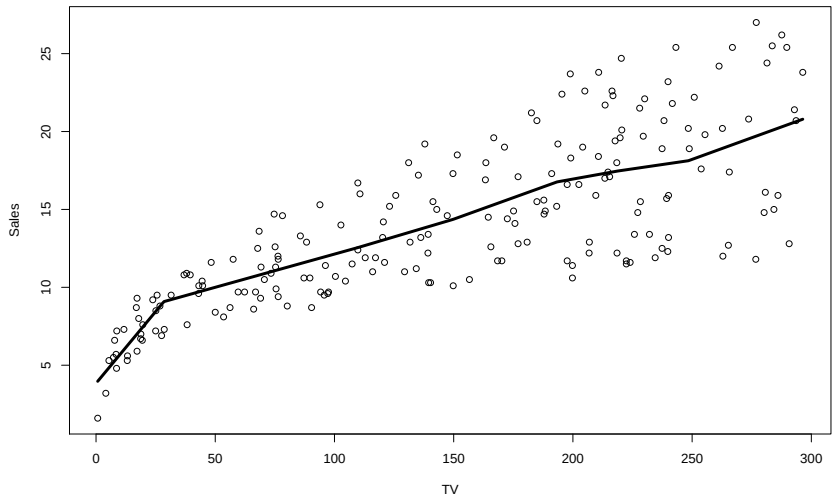
4. A cubic spline with K knots.

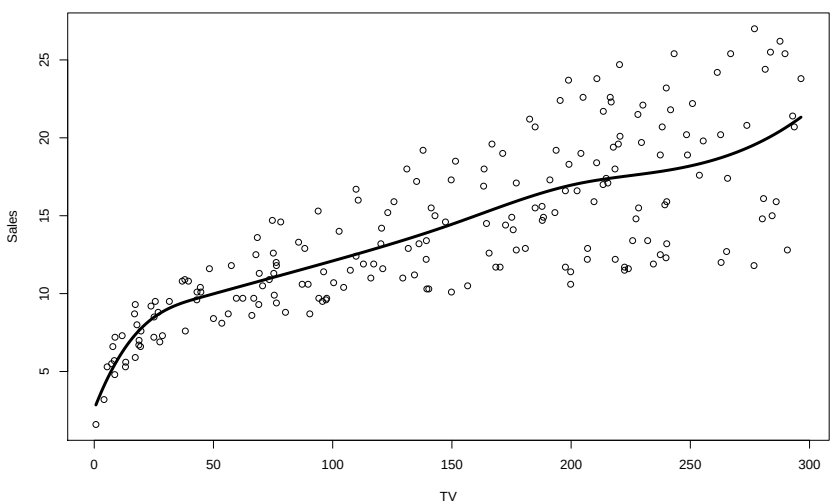
5. A natural cubic spline with K knots.

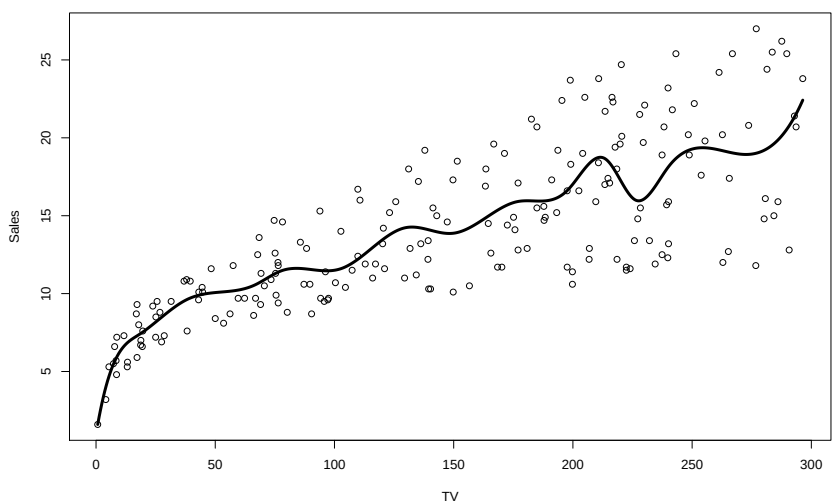
Example: different methods predicting Sales from TV

Plots below show step functions (which #intervals?), a spline of degree 1 and 8 df, a spline of degree 3 and 8 df and a spline of degree 3 and 16 df. Which one is which?









Fitting splines in R

One implementation: `splines` package.

Polynomial splines: function **bs(x,...)**. Arguments:

- **degree**: degree of polynomial, values $1, 2, \ldots$
- **knots**: specify knots as discussed in lectures.
- **df**: specify degrees of freedom.
- Only one of the arguments **df** or `knots` should be specified at a time.
- If **df** is specified knots will be put at quantiles of predictors.

Natural cubic splines: function **ns(x,...)**. Arguments:

- **knots**: specify knots as discussed in lectures.
- **df**: specify degrees of freedom.
- Only one of the arguments **df** or **knots** should be specified at a time.
- If **df** is specified knots will be put at quantiles of predictors.

Both functions output a matrix with predictors that can be used in function **lm**.

Counting degrees of freedom: by default, no intercept is included. Thus degrees of freedom are counted without intercept, so 1 less than discussed in lectures.

Basis expansions: closing remarks

All the methods we discussed so far (step functions, piecewise polynomials, polynomial splines, natural splines) can be formulated as liner models with new predictors that are obtained from non-linear transformations of predictor x, i.e. a model of the form

$$f(x) = b_0 + b_1 g_1(x) + b_2 g_2(x) + ... + b_{d+1} g_d(x)$$

where $g_1, ..., g_d$ are fixed functions. This is sometimes called *basis expansion*. We discussed several choices of functions $g_1, ..., g_d$, many more choices exist.

- Some popular approaches we did not cover: wavelets and Fourier series.

- A common problem is all examples above: we need to pick the location of knots/breakpoints.
- Automatic way: pick according to quantiles of predictors, or uniformly spaced over range of predictors. This does not always work well... This motivates smoothing splines discussed next.

Exercise: write down the functions $g_1, ..., g_d$ for the following cases: polynomial regression of degree 4, step functions with intervals $(0, 1], (1, 3], (3, 4]$, piecewise polynomial regression of degree 3 with intervals $(0, 2], (2, 4]$, piecewise polynomial regression of degree 1 with intervals $(0, 2], (2, 4]$, cubic spline with knots $1, 2, 5$.

Some R examples: file `Sta314-09-Lecture07-BasisExpansions.R`

Smoothing splines

Smoothing splines

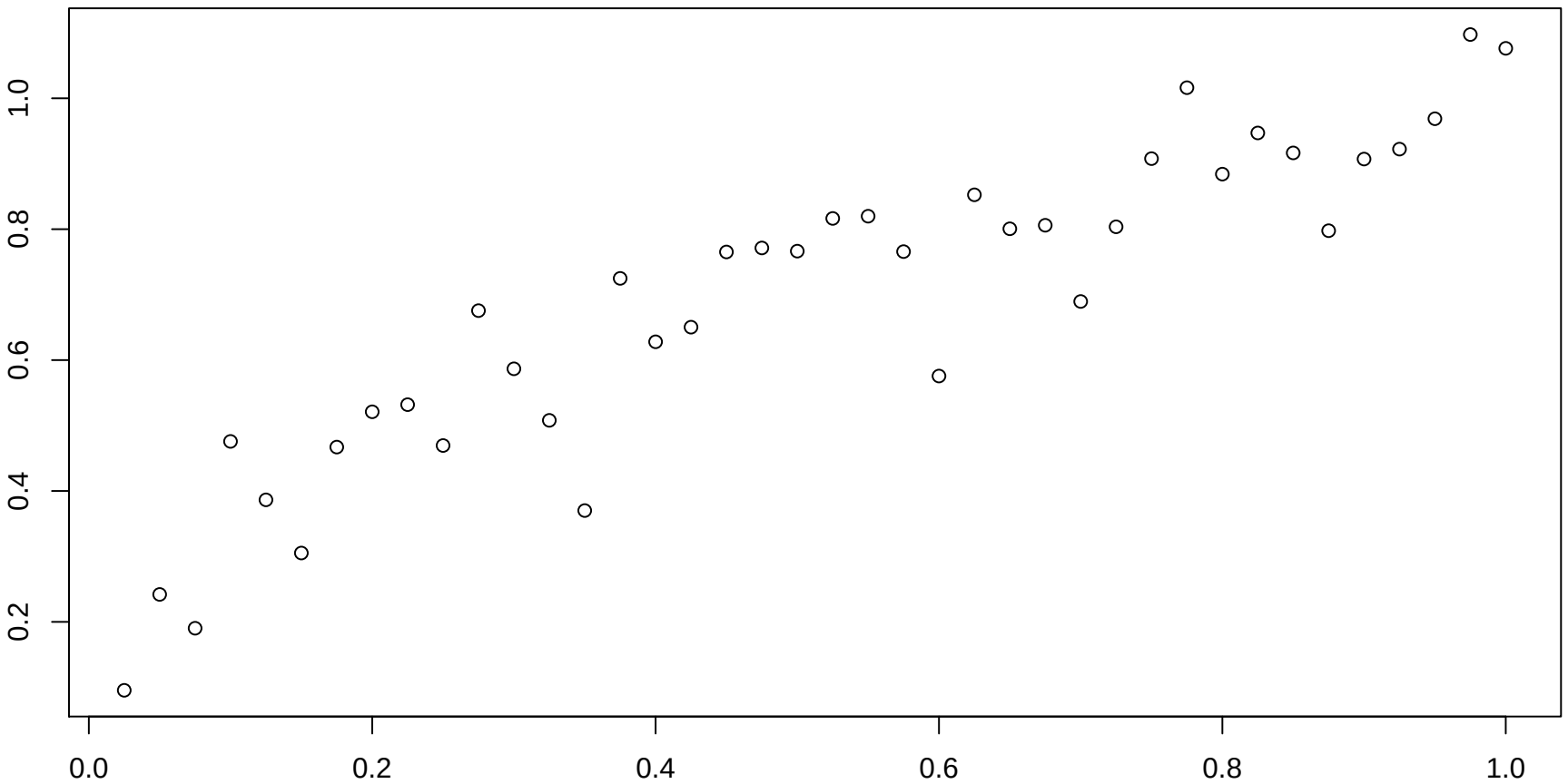
Smoothing splines avoid need to choose location of knots manually. Flexibility of smoothing splines is controlled by a 'penalty parameter', similar to ridge and lasso.

Given data $(x_i, y_i)_{i=1,\ldots,n}$, the corresponding smoothing spline $\hat{f}$ is defined as

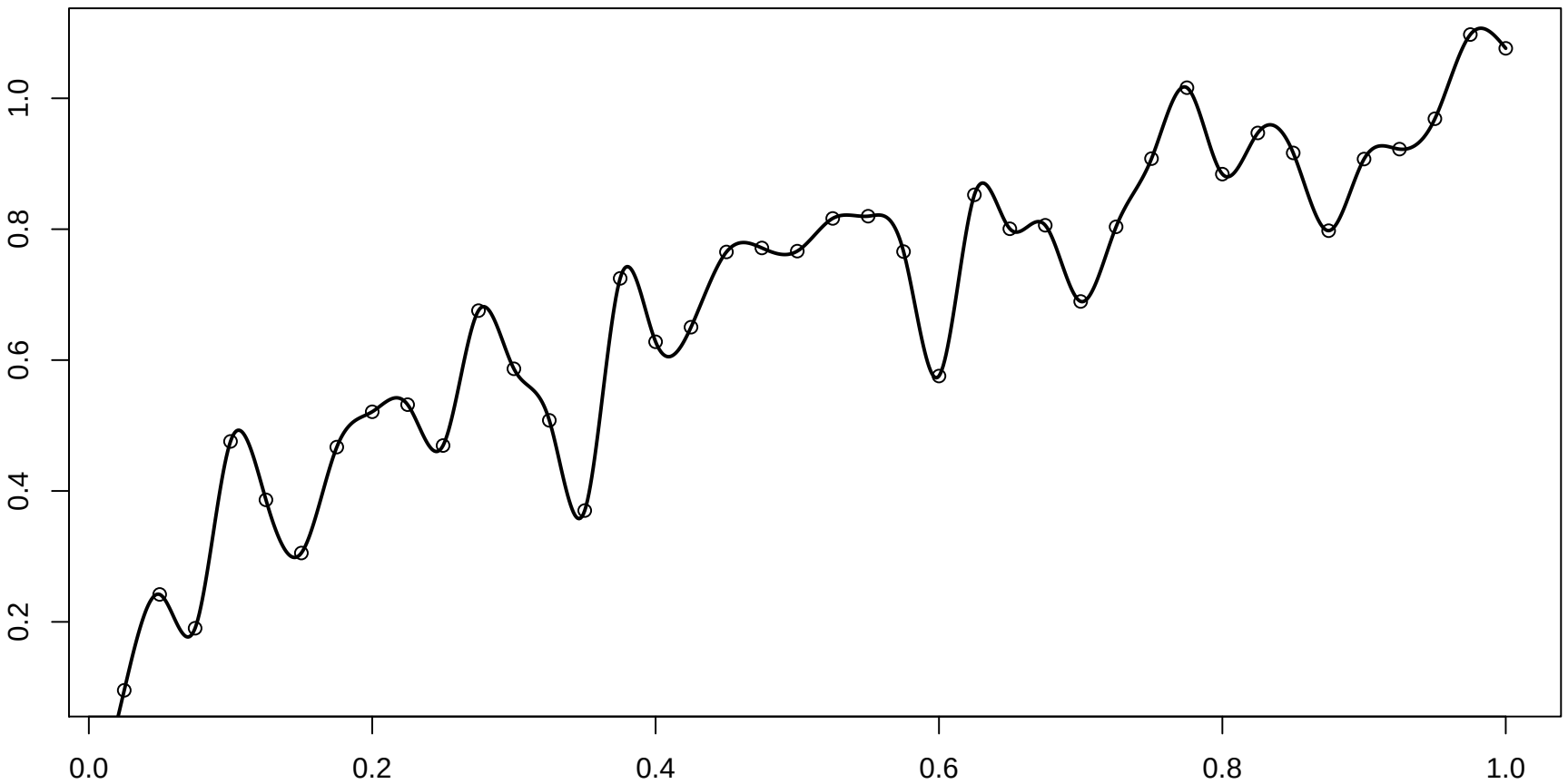
$$\hat{f} := \arg\min_{g \in \mathcal{C}^2} \left\{ \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int (g''(x))^2 dx \right\}$$

- $\mathcal{C}^2$ denotes the space of all functions $\mathbb{R} \to \mathbb{R}$ which have two continuous derivatives (smooth functions).
- $g''(x)$ is the second derivative of g in point x.
- The first part $\sum_{i=1}^{n} (y_i - g(x_i))^2$ corresponds to the training MSE when predicting y_i with $g(x_i)$. This is similar as RSS for linear models.
- Question 1: why not just minimize $\sum_{i=1}^{n} (y_i - g(x_i))^2$? What will happen?

Why not just minimize $\sum_{i=1}^{n}(y_i - g(x_i))^2$?

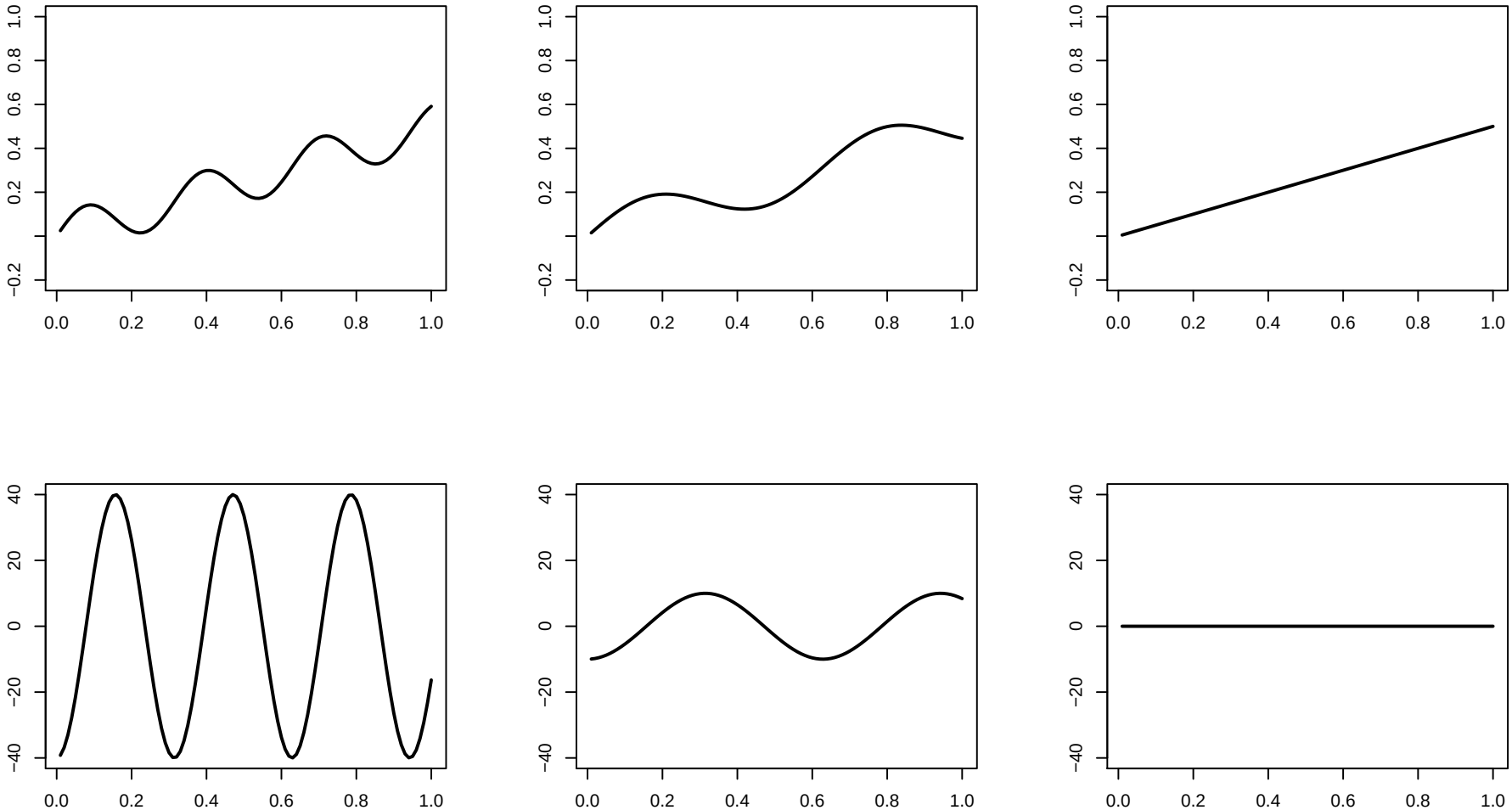


Why not just minimize $\sum_{i=1}^{n}(y_i - g(x_i))^2$?



What is the effect of $\int (g''(x))^2 dx$?

Plots of g (top row) and g'' (bottom row) for some examples of functions g (scale on y axis same for all 3 plots in bottom row). More 'wiggles' in g mean larger g''.



- Left to right: $\int (g''(x))^2 dx \approx 814,\ 52,\ 0.$
- Adding $\lambda \int (g''(x))^2 dx$ to $\sum_{i=1}^{n} (y_i - g(x_i))^2$ will prevent g from overfitting.

Smoothing splines: continued

Theorem For any $\lambda > 0$ the function $\hat{f}$ defined by

$$\hat{f} := \arg\min_{g \in C^2} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int (g''(x))^2 dx$$

is *natural cubic spline with knots at* $x_1, ..., x_n$!

Hence $\hat{f}$ can be represented in the form

$$\hat{f}(x) = \sum_{k=1}^{n} \hat{b}_k g_k(x)$$

for some basis functions $g_1, ..., g_n$ (which depend on data, we omit the details). For computing $\hat{f}$: we just need to maximize over a finite number of parameters!

Advantage of smoothing splines over basis expansion: in smoothing splines, we only need to select the parameter λ (positive number), the location and number of knots is selected automatically! The parameter λ can be selected by cross-validation.

- There is a way to specify 'degrees of freedom' for smoothing splines, we omit the details here (ask me after lectures if you are interested).

Local polynomial regression

Local polynomial regression

Degree d, *kernel* $K : [0,1] \to \mathbb{R}$, *span* $s \in (0,1)$ computed at x_0.

---

1. Let k be integer part of ns. Select k points among $x_1, ..., x_n$ that are closest to x_0, denote those by $z_1, ..., z_k$, and denote corresponding response values by $y_1^*, ..., y_k^*$. Let $m := \max\{|x_0 - z_1|, ..., |x_0 - z_k|\}$
2. Consider the minimization problem

$$\hat{B} := \arg \min_{(b_1,...,b_{d+1})^\top \in \mathbb{R}^{d+1}} \sum_{i=1}^{k} K(|x_0 - z_i|/m)(y_i^* - b_1 - b_2 z_i - ... - b_{d+1} z_i^d)^2$$

3. Regression function at x_0 is of the form $\hat{f}(x_0) = \hat{b}_1 + x_0 \hat{b}_2 + ... + \hat{b}_{d+1} z_i^d$.

Special case: $d = 0$ (local constant regression). Let $W_i(x_0) := K(|x_0 - z_i|/m)$. Then (see blackboard for derivation)

$$\hat{f}(x_0) = \hat{b}_1 = \frac{\sum_{i=1}^{k} W_i(x_0) y_i^*}{\sum_{i=1}^{k} W_i(x_0)},$$

i.e. *weighted average* of y_i corresponding to k nearest points.

Local constant regression: intuition

Special case: $d = 0$ (local constant regression). Let $W_i(x_0) := K(|x_0 - z_i|/m)$. Then (see blackboard for derivation)

$$\hat{f}(x_0) = \hat{b}_1 = \frac{\sum_{i=1}^k W_i(x_0) y_i^*}{\sum_{i=1}^k W_i(x_0)},$$

i.e. *weighted average* of y_i corresponding to k nearest points.

Assume that $y_i = f(x_i) + \varepsilon_i$, $\mathbb{E}[\varepsilon_i] = 0$, $Var(\varepsilon_i) = \sigma^2$. Then

$$\mathbb{E}[\hat{f}(x_0)] - f(x_0) = \frac{\sum_{i=1}^k W_i(x_0) f(z_i)}{\sum_{i=1}^k W_i(x_0)} - f(x_0) = \sum_{i=1}^k \frac{W_i(x_0)}{\sum_{\ell=1}^k W_\ell(x_0)} (f(z_i) - f(x_0))$$

$$Var(\hat{f}(x_0)) = \sigma^2 \sum_{i=1}^k \left(\frac{W_i(x_0)}{\sum_{\ell=1}^k W_\ell(x_0)} \right)^2$$

- Can try to balance bias and variance by choice of kernel K.
- Since $f(z_i) - f(x_0)$ larger if $|z_i - x_0|$ larger give less weight to such observations.
- Intuition for local polynomial regression of arbitrary orders similar.

Local polynomial regression: motivation

What can we gain by considering local polynomial regression with degree > 0?

Boundary effects: see picture on blackboard and simulation

General motivation: Taylor expansion. If a function f is d times continuously differentiable at x_0, then by Taylor expansion

$$f(z) = f(x_0) + f^{(1)}(x_0)(z - x_0) + \ldots + \frac{1}{d!} f^{(d)}(x_0)(z - x_0)^d + r(x_0, z)$$

where $|r(x_0, z)|/|x_0 - z|^d \to 0$ for $z \to x_0$.

- The blue part is a polynomial in z.
- The red part is a remainder, 'small' if $|x_0 - z|$ small.
- Local polynomial regression tries to model the blue part.

Local polynomial regression: closing comments

- For proper choice of kernel K (i.e. $K(1) = 0$, K continuous) this leads to functions $\hat{f}$ which are continuous. Advantage compared to k-nn.
- Typical choice of degree d is $d = 1$, sometimes $d = 3$.
- There are many variations of 'local weighting idea', for example using all points in a local window of given length.
- One of many **R** implementations: `loess` function in `splines` package. Uses kernel $K(x) = (1 - x^3)^3$.
- Replacing $|x_0 - z_i|$ by other way of measuring distances (e.g. Euklidean distance), this can be extended to predictors with values in $\mathbb{R}^p$. Typically doe snot work well beyond about $p = 3$. This is because of the 'curse of dimensionality'.

Some R examples: file `Sta314-09-Lecture07-SmoothingSplinesLocalregr.R`