`richardwu.ca`

STAT 444/844 Course Notes

Statistical Learning: Function Estimation

Kun Liang • Winter 2019 • University of Waterloo

Last Revision: March 26, 2019

Table of Contents

Abstract

These notes are intended as a resource for myself; past, present, or future students of this course, and anyone interested in the material. The goal is to provide an end-to-end resource that covers all material discussed in the course displayed in an organized manner. These notes are my interpretation and transcription of the content covered in lectures. The instructor has not verified or confirmed the accuracy of these notes, and any discrepancies, misunderstandings, typos, etc. as these notes relate to course's content is not the responsibility of the instructor. If you spot any errors or would like to contribute, please contact me directly.

1 January 8, 2019

1.1 What is a function?

Suppose we have some measured **response** variate y and we have one or more **explanatory** variables $x_1, \ldots, x_p$. The response and explanatory ariables are approximately related through an unknown function $\mu(x)$ (to be estimated/learned) where

$$y = \mu(x) + r$$

where r is residual that cannot be explained by $\mu(x)$.
Some other names for response and explanatory variables include:

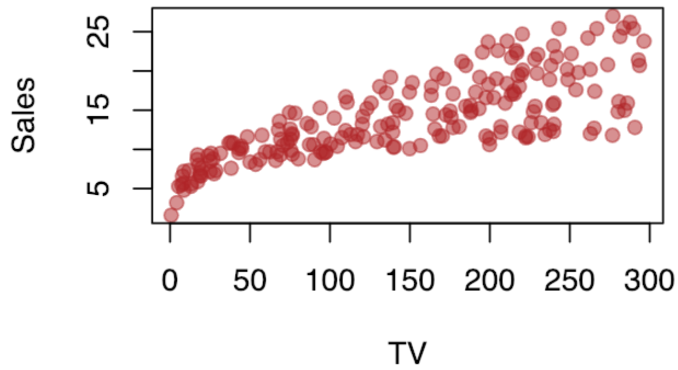
response	explanatory
response	predictor
response	design
output	input
dependent	independent
endogenous	exogenous

1.2 Advertising data example

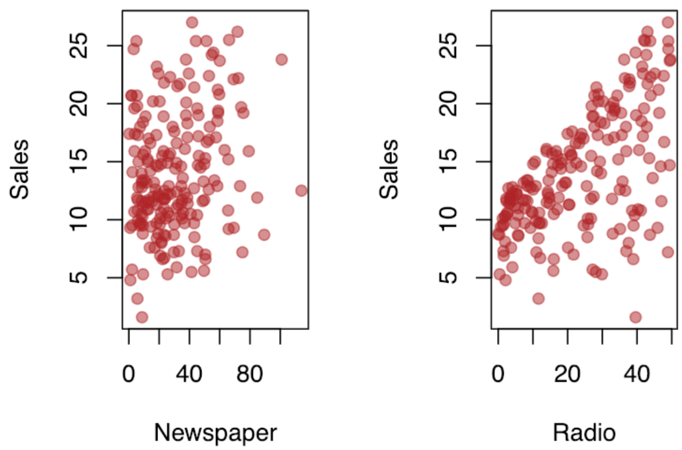
Suppose we want to predict Sales (response) from how much companies spend on TV, Radio, and Newspaper advertising (explanatory).
The dataset is

```
##   X    TV Radio Newspaper Sales
## 1 1 230.1  37.8      69.2  22.1
## 2 2  44.5  39.3      45.1  10.4
## 3 3  17.2  45.9      69.3   9.3
## 4 4 151.5  41.3      58.5  18.5
## 5 5 180.8  10.8      58.4  12.9
## 6 6   8.7  48.9      75.0   7.2
```

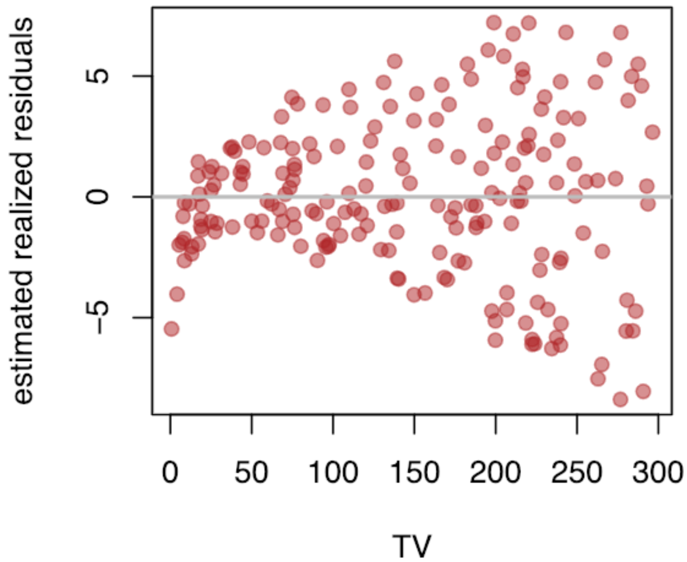
if we plot sales against TV



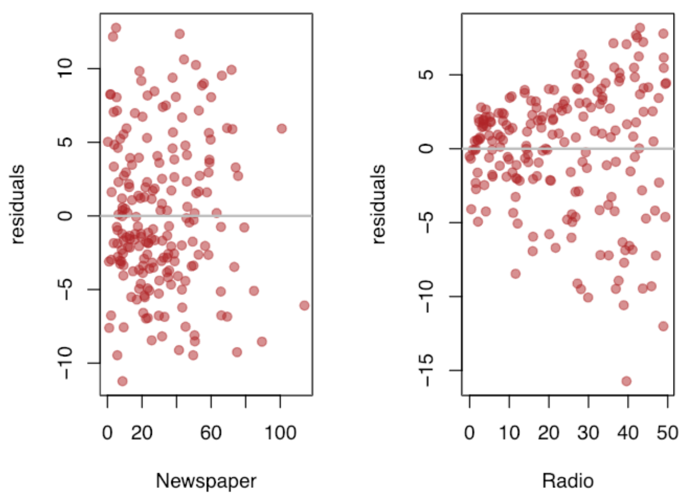
we see there is some positive correlation.
Similarly against Newspaper and Radio



What if we tried a simple linear model where $\hat{\mu}(x_1) = \hat{\alpha} + \hat{\beta}x_1$ where x_1 is the TV advertising? We obtain estimates $\hat{\alpha} = 7.03$ and $\hat{\beta} = 0.05$ which are interpretable. However if we take a look at the residuals



we see that the residuals are not independently distributed accordingly to x_1, which violates our Markov-Gaussian assumptions.
The residuals of the model with Newspaper and Radio are



we observe that we do not observe constant variance across the explanatory variables.
Therefore a linear model does not seem to work (we could of course introduce scaling e.g. log-scaling for the Radio variate or polynomial terms).

1.3 Notation

Some notes on notation:

- Capital letters are matrices or vectors: A, X, Σ
- Lower letters are scalars: a, x, σ

- Arrows on letters are vectors: $\vec{a}, \vec{x}$
- All vectors are column vectors
- The transpose of any matrix A is A^T (ocassionally A')

1.4 Definitions and properties

Quadratic form Suppose $A = (a_{ij})_{n\times n}$ is symmetric i.e. $a_{ij} = a_{ji}\ \forall i, j$. Then

$$\begin{aligned} f &= Y^T AY \\ &= \sum_i \sum_j a_{ij} y_i y_j \end{aligned}$$

is called a **quadratic form**.

Trace For a matrix, the **trace** is defined as

$$\text{tr}(A_{m\times m}) = \sum_{i=1}^{m} a_{ii}$$

Note that $\text{tr}(BC) = \text{tr}(CB)$.

Rank The **rank** of a matrix denoted $\text{rank}(A)$ is the maximum number of *linearly independent columns* (or rows) of A.

Note that vectors $Y_1, \ldots, Y_n$ are linearly independent iff

$$c_1Y_1 + \ldots + c_nY_n = 0$$

implies $c_1 = \ldots = c_n = 0$ (i.e. no non-trivial solution).

Eigenvector and eigenvalue A non-zero vector $\vec{v}_i$ is an **eigenvector** of $A_{m\times m}$ if

$$A\vec{v}_i = \lambda_i \vec{v}_i \qquad i = 1, 2, \ldots, m$$

where λ_i is the corresponding ith **eigenvalue**.

Idempotent A matrix A is **idempotent** if $AA = A$.

Some notable results:

1. If A is idempotent, then all its eigenvalues are either 0 or 1.
2. If A is idempotent, there exists an orthogonal matrix P such that $A = P\Lambda P^T$ where

$$\Lambda = \begin{bmatrix} 1 & \ldots & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & \ldots & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix}$$

and $\text{tr}(A) = \text{rank}(A) = \text{tr}(\Lambda)$ which is equivalent to the number of eigenvalues being 1.

2 January 10, 2019

2.1 Linear models

A linear model is generally in the form of

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \epsilon_i \qquad i = 1, \ldots, n$$

which holds under the assumptions that

- $E(\epsilon_i) = 0$
- $Var(\epsilon_i) = \sigma^2$ (constant variance)
- $\epsilon_1, \ldots, \epsilon_n$ are independent
- $\epsilon_1, \ldots, \epsilon_n \overset{iid}{\sim} N(0, \sigma^2)$

In matrix form we have

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}_{n\times 1} = \begin{bmatrix} 1 & x_{11} & \ldots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \ldots & x_{np} \end{bmatrix}_{n\times(p+1)} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}_{(p+1)\times 1} + \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}_{n\times 1}$$

or in short matrix form $Y = X\vec{\beta} + \vec{\epsilon}$.
The **Least Squares Estimator (LSE)** of $\vec{\beta}$ minimizing the discrepancy function

$$S(\vec{\beta}) = (Y - X\vec{\beta})^T (Y - X\vec{\beta})$$

has a closed form solution

$$\hat{\vec{\beta}} = (X^T X)^{-1} X^T Y$$

The fitted values are thus

$$\begin{aligned} \hat{Y} = X\hat{\vec{\beta}} &= X(X^T X)^{-1} X^T Y \\ &= HY \end{aligned}$$

where $H = X(X^T X)^{-1} X^T$ (**hat matrix**). Note that H is **idempotent** and **symmetric**.
Geometric interpretation of LSE: $\hat{Y}$ is the projection of Y onto $C(X)$, the column space of X (we can thus see that the fitted errors should be orthogonal to our fitted values in LSE).
The **degrees of freedom** of our model is $n - (p + 1)$ where $p + 1$ is the number of free parameteres in our model. This is equivalent to $n - \mathrm{tr}(H)$ i.e. $\mathrm{tr}(H) = p + 1$.
Under normality

- $\hat{\vec{\beta}} = MVN(\vec{\beta}, \sigma^2 (X^T X)^{-1})$
- $\hat{\vec{\beta}}$ and $\hat{\sigma}^2$ are independent (Note $\hat{\sigma}^2 = \frac{SSE}{df}$).
- $\frac{(n-p-1)\hat{\sigma}^2}{\sigma^2} \sim \chi^2_{n-p-1}$

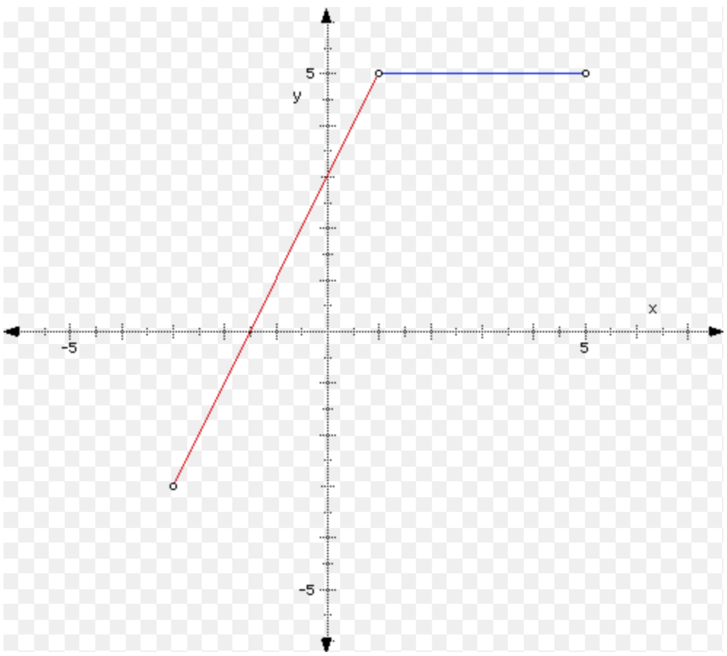
Let $\vec{a}_p = (1, x_1, \ldots, x_p)^T$ (observation $\vec{x}$ extended with intercept term). The $(1-\alpha)$ **prediction interval** at $\vec{a}_p$ is

$$\vec{a}_p^T \hat{\vec{\beta}} \pm t_{n-p-1,\alpha/2} \hat{\sigma} \sqrt{1 + \vec{a}_p^T (X^T X)^{-1} \vec{a}_p}$$

We can also estimate **confidence intervals** as well (drop $1 + \ldots$ term above).

2.2 Piecewise linear

We can specify the following piecewise linear function (with discontinuity at a)



as two linear functions

$$y = \begin{cases} \beta_0 + \beta_1 x & x \leq a \\ \beta_2 + \beta_3 x & x \geq a \end{cases}$$

subject to $\beta_0 + \beta_1 a = \beta_2 + \beta_3 a$.
A more convenient way to express the above

$$y = \beta_0 + \beta_1 x + \beta_2 (x - a) I(x \geq a)$$

where I is the indicator function. Note the above is linear in terms of $\vec{\beta}$ BUT NOT in terms of x. However we can simply construct a new variate $(x - a)I(x \geq a)$ from x.
Note that β_2 is the *change in slope right of a* for samples where $x \geq a$.
Extension to more than one interesting point (knot) is straightforward.

2.3 Piecewise quadratic

Similar to piecewise linear models, we can specify

$$y = \begin{cases} \beta_0 + \beta_1 x + \beta_2 x^2 & x \leq a \\ \beta_3 + \beta_4 x + \beta_5 x^2 & x \geq a \end{cases}$$

subject to $\beta_0 + \beta_1 a + \beta_2 a^2 = \beta_3 + \beta_4 a + \beta_5 a^2$ (continuity) and $\beta_1 + 2\beta_2 a = \beta_4 + 2\beta_5 a$ (differentiable at a). Alternatively we can express this as one linear function

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 (x-a)^2 I(x \geq a)$$

continuity is trivially satisfied. Note the 1st derivative is

$$\frac{dy}{dx} = \beta_1 + \beta_2 x + 2\beta_3 (x-a) I(x \geq a)$$

where the last term is 0 when $x = a$, thus our additional indicator term does not affect the derivative.

Remark 2.1. We choose to omit the $(x-a)I(x \geq a)$ term to ensure y is differentiable at $x = a$.

2.4 Weighted least squares

Sometimes we would like to give more importance to some observations than others.
Instead of minimizing $(Y - X\vec{\beta})^T(Y - X\vec{\beta})$ we can minimize

$$(Y - X\vec{\beta})^T W (Y - X\vec{\beta})$$

where

$$W = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & w_n \end{bmatrix}_{n \times n}$$

a diagonal matrix. w_i corresponds to the weight assigned to observation i (a higher w_i the more important that observation is).

Claim. The closed form solution is

$$\hat{\vec{\beta}}_{WLS} = (X^T W X)^{-1} X^T W Y$$

Proof. Note that

$$\begin{aligned} S(\vec{\beta}) &= (Y - X\vec{\beta})^T W (Y - X\vec{\beta}) \\ &= Y^T W Y - Y^T W X \vec{\beta} - \vec{\beta}^T X^T W Y + \vec{\beta}^T X^T W X \vec{\beta} \end{aligned}$$

Note that $Y^T W X \vec{\beta} = (\vec{\beta}^T X^T W Y)^T$ which is a scalar, so $Y^T W X \vec{\beta} = \vec{\beta}^T X^T W Y$ (transposes of scalars are equivalent). thus

$$S(\vec{\beta}) = Y^T W Y - 2Y^T W X \vec{\beta} + \vec{\beta}^T X^T W X \vec{\beta}$$

where $-2Y^T W X \vec{\beta}$ is the "linear term" and $\vec{\beta}^T X^T W X \vec{\beta}$ is of quadratic form.

Recall that

$$\frac{d\vec{c}^T Y}{dY} = \vec{c}^T$$
$$\frac{dY^T AY}{dY} = 2Y^T A$$

so

$$\begin{aligned}
&\frac{dS(\vec{\beta})}{d\vec{\beta}} = -2Y^T WX + 2\vec{\beta}^T X^T WX \\
\Rightarrow&\vec{\beta}^T X^T WX = Y^T WX && \frac{dS(\vec{\beta})}{d\vec{\beta}} = 0 \\
\Rightarrow&(X^T WX)\vec{\beta} = X^T WY && W^T = W \\
\Rightarrow&\vec{\beta} = (X^T WX)^{-1} X^T WY
\end{aligned}$$

as claimed. □

Here is an alternative proof:

Proof. Let $Y^* = W^{\frac{1}{2}} Y$ and $X^* = W^{\frac{1}{2}} X$.
Note that minimizing $(Y - X\vec{\beta})^T W(Y - X\vec{\beta})$ is equivalent to minimizing $(Y^* - X^*\vec{\beta})^T (Y^* - X^*\vec{\beta})$ (simply expand out X^* and Y^*).
Thus the LSE of $\vec{\beta}$ with X^*, Y^* is

$$\begin{aligned}
\vec{\beta} &= ({X^*}^T X)^{-1} {X^*}^T Y^* \\
&= ((X^T W^{\frac{1}{2}})(W^{\frac{1}{2}} X))^{-1} (X^T W^{\frac{1}{2}})(W^{\frac{1}{2}} Y) \\
&= (X^T WX)^{-1} X^T WY
\end{aligned}$$

which is equivalent to our previous derivation. □

3 January 15, 2019

3.1 Weight least squares applications

Example 3.1. We can apply weighted least squares to do **local regression** where we downweight observations farther away from a given observation.

Example 3.2. Suppose that $Var(\epsilon_i) = \sigma_i^2$ (i.e. not all observations are drawn with the same variance). If we want to overweight observations that have lower variance, we can set $w_i = \frac{1}{\sigma_i^2}$ to obtain an **unbiased estimator of $\vec{\beta}$** with the *smallest variance* (**Best Linear Unbiased Estimator** or **BLUE**).

3.2 Types of errors

We will use the example of 790 *Facebook* posts published by a comestics company to illusrate.
The population being examined is called the **study population** (the 790 posts). The analysis of these posts may be applied to a larger population (whether it's future Facebook posts for this company or Facebook posts for *any* company) which we call the **targt population**.
The difference between the study and target population is called the **study error**.

In a paper by Soros et al., they ended up using a **sample** of only 500 posts for confidentiality reasons. The difference between the sample and study population is called **sample error**.

4 January 17, 2019

4.1 Notes on terminology and `lm` in R

- When using `lm` the intercept term is included by default. To remove it simply specify `Y ~ X - 1`.
- **Factors** are like categorical variables in R: there are a finite number of categories (called **factor levels**).
- In `lm` almost any function of variates may appear in the formula e.g. `Y ~ X + sin(X)` or `Y ~ X + sin(X * Y)`.

  To specify $Y = X \cdot Z$, we need to use `Y ~ I(X * Z)` or `Y ~ X:Z` *instead of* `X * Z` since `X * Z` represents **interaction** in `lm` and translates to the model $y = \alpha x + \beta z + \gamma xz + r$.
- Some arithmetic operations e.g. $+, -, *, \hat{}$ are interpreted as formula operators rather than arithmetic operators in `lm`. One should wrap them in `I(·)`.

4.2 Notes on model selection

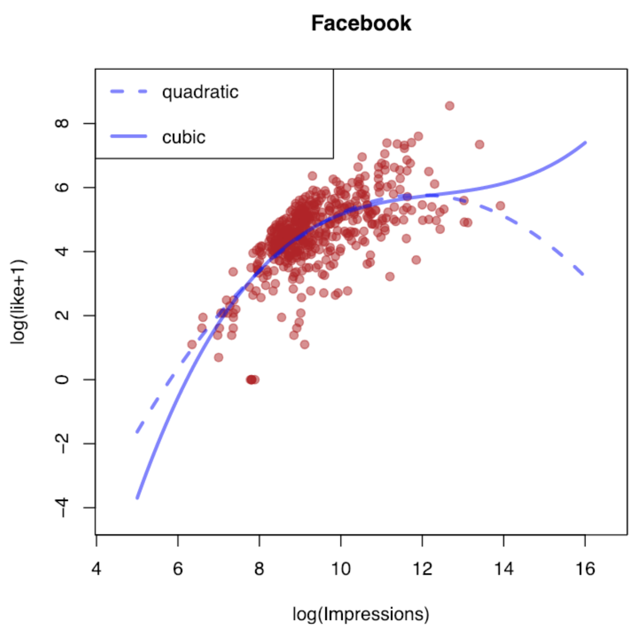


Figure 4.1: Quadratic and cubic polynomial linear models on Facebook data.

In the above figure we see that while both quadratic and cubic models are **global** (predict any value of x) the quadratic model seems to predict likes returning to 0 as impressions approach infinity.
The cubic function on the contrary continues to increase: this makes more sense intuitively, thus examining a model often requires human understanding of the data and problem.

4.3 Geometric interpretation of linear models

A linear model is a linear combination of functions called **generators** e.g.

$$\mu(x) = \beta_1 g_1(x) + \beta_2 g_2(x) + \beta_3 g_3(x) + \beta_4 g_4(x)$$

where $g_1, \ldots, g_4$ could be arbitrary continuous functions of x.
All possible linear combinations of the generators forms a **subspace** (the functions *generate* the subspace). The functions are a **basis** for this subspace. $\mu(x)$ lies in the subspace whose dimension equals to the number of basis functions.
The functions should be *linearly independent* of each other: otherwise the solution to parameters will be ill-defined.

5 January 24, 2019

5.1 Discrepancy function

Let the **discrepancy function** for a fit of parameters $\vec{\beta}$ be denoted

$$S(\vec{\beta}) = \sum_{i=1}^{n} \rho(y_i - \vec{x}_i^T \vec{\beta}) = \sum_{i=1}^{n} \rho(r_i)$$

where ρ is a real-valued **loss function** (in the OLS case, this was simply the square function). r_i is our residual for observation i.
Taking the derivative

$$\begin{aligned} \frac{dS(\vec{\beta})}{d\vec{\beta}} &= \sum_{i=1}^{n} \rho'(y_i - \vec{x}_i^T \vec{\beta})(-1)\vec{x}_i^T \\ &= -\sum_{i=1}^{n} \rho'(r_i)\vec{x}_i^T \end{aligned}$$

Solving for the extremum we have

$$\sum_{i=1}^{n} \psi(r_i)\vec{x}_i^T = \vec{0}^T$$

where $\psi(r) = \rho'(r)$ (derivative with respect to $\vec{\beta}$).

Remark 5.1 (LSE)**.** If $\rho(r) = r^2$ we get LSE, that is ($\psi(r) = 2r$)

$$\begin{aligned} \vec{0} &= \sum_{i=1}^{n} 2r\vec{x}_i \\ &= 2\sum_{i=1}^{n} (\vec{x}_i y_i - \vec{x}_i^T \vec{x}_i \vec{\beta}) \\ &= 2(X^T Y - X^T X \vec{\beta}) \end{aligned}$$

which exactly solves to our LSE closed form solution.

5.2 Discrepancy function and log-likelihood

Let us compare our discrepancy function with the log-likelihood for linear models:

$$
\begin{aligned}
l(\vec{\beta}) &= \sum_{i=1}^{n} l_i(\vec{\beta}) \\
&= \sum_{i=1}^{n} l(r_i)
\end{aligned}
$$

where $l(r_i) = \frac{-r_i^2}{2\sigma^2}$, a function only of r_i.
The second equality follows from the following remark:

Remark 5.2. Note $l_i(\vec{\beta})$ is the ith observation's contribution to $l(\vec{\beta})$ i.e.

$$l_i(\vec{\beta}) = \log f(y_i \mid \vec{x}_i^T \vec{\beta}, \sigma^2)$$

For a linear model we have

$$f(y_i \mid \vec{x}_i^T \vec{\beta}, \sigma^2) \sim N(\vec{x}_i^T \vec{\beta}, \sigma^2)$$

so $l_i(\vec{\beta}) = -\frac{r_i^2}{2\sigma^2} + C$ where $C = -\frac{1}{2}\log(2\pi) - \log\sigma$ a constant.
We let $l(r_i) = -\frac{r_i^2}{2\sigma^2}$. Since the constant does not change with respect to β we can omit it from our objective function.

From above we observe that minimizing the discrepancy function is the same as maximizing the log likelihood where $\rho(r) = -l(r)$ in the discrepancy function.

Definition 5.1 (M-estimator)**.** We call the estimator $\vec{\beta}$ that minimizes $\sum_{i=1}^{n} \rho(r_i)$ the **M-estimator** or the **maximum-likelihood type estimator**.

5.3 Iteratively re-weighted least squares (IRLS)

Note that the solution turns out to be a *WLS estimator*:

$$
\begin{aligned}
\vec{0} &= \sum_{i=1}^{n} \psi(r_i)\vec{x}_i \\
&= \sum_{i=1}^{n} \frac{\psi(r_i)}{r_i} r_i \vec{x}_i \\
&= \sum_{i=1}^{n} w(r_i)(y_i - \vec{x}_i^T\vec{\beta})\vec{x}_i \\
&= \sum_{i=1}^{n} w_i(y_i - \vec{x}_i^T\vec{\beta})\vec{x}_i
\end{aligned}
$$

where we let $w_i = w(r_i) = \frac{\psi(r_i)}{r_i}$. If we solve this we see that the solution is WLS where

$$\hat{\vec{\beta}} = (X^TWX)^{-1}X^TWY$$

with $W = \text{diag}(w_1, \ldots, w_n)$.

However, the weights of this WLS depend on the residuals which in turn depends on $\vec{\beta}$. If we are given an initial estimate of $\vec{\beta}^{(0)}$, we could *iteratively update residuals* and $\vec{\beta}$ to converge to a solution. We proceed as follows:

Initialization Initialization: set $j = 0$

Step 1 Compute residuals

$$\hat{r_i}^{(j)} = y_i - \vec{x}_i^T \hat{\vec{\beta}}^{(j)} \qquad i = 1, \ldots, n$$

Step 2 Update weights

$$w_i^{(j)} = \frac{\psi(\hat{r}_i^{(j)})}{\hat{r}_i^{(j)}}$$

and let $W^{(j)} = \text{diag}(w_1^{(j)}, \ldots, w_n^{(j)})$.

Step 3 WLS to estimate next set of $\hat{\vec{\beta}}^{(j+1)}$

$$\hat{\vec{\beta}}^{(j+1)} = (X^T W^{(j)} X)^{-1} X^T W^{(j)} Y$$

Step 4 Set $j = j + 1$ and return to Step 1 if convergence criterion is not met.

We can this procedure **iteratively re-weighted least squares (IRLS)**.
The convergence criterion is typically

$$\|\hat{\vec{\beta}}^{(j+1)} - \hat{\vec{\beta}}^{(j)}\| \leq \epsilon$$

with the L2/Euclidean norm and for some small positive constant ϵ.

5.4 Why IRLS?

Question 5.1. Why do we need to use iteratively re-weighted least squares?
In ordinary least squares with Gaussian response and loss function r_i^2, there is no reason to use IRLS since OLS and IRLS are equivalent (the loss function r_i^2 simplifies IRLS to OLS).
However in **generalized linear models (GLMs)** (STAT 431/831) we may have a different type of response (e.g. Bernoulli 0/1 or categorical) and thus we may define our loss function $\rho(r_i)$ differently.
We may also want to modify our $\rho(r_i)$ to de-emphasize huge outliers (see next section).

5.5 Robust regression

Robust regression tries to de-emphasize the influence of large outliers.

Question 5.2. What loss function ρ (and ψ) should we use?
In ordinary least squares (OLS) we can use

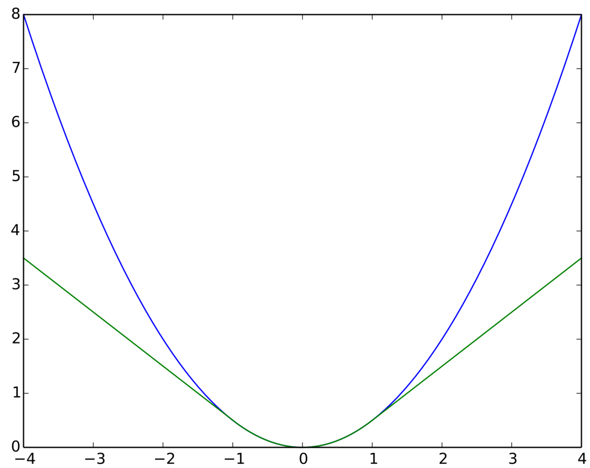
$$\begin{aligned} \rho(r) &= \frac{1}{2} r^2 \\ \psi(r) &= r \\ w(r) &= \frac{\psi(r)}{r} = 1 \end{aligned}$$

so we essentially have $W = I_{n \times n}$ which devolves into OLS as expected.

Remark 5.3. The residual function for OLS is unbounded and so extreme outliers with large residuals have significantly more influence.
Huber (1964) proposed a modified loss function (**Huber loss**) which de-emphasizes outliers:

$$\rho(r) = \begin{cases} \frac{1}{2}r^2 & \text{if } |r| \leq c \\ c(|r| - \frac{1}{2}c) & \text{if } |r| > c \end{cases}$$

The modified loss function essentially makes the loss function linear after a certain threshold c:



We also let

$$\psi(r) = \begin{cases} r & \text{if } |r| \leq c \\ c\text{sign}(r) & \text{if } |r| > c \end{cases}$$

and thus

$$w(r) = \begin{cases} 1 & \text{if } |r| \leq c \\ \frac{c}{|r|} & \text{if } |r| > c \end{cases}$$

The ψ and weight w functions look like

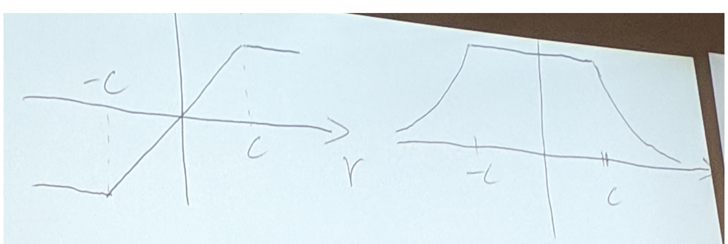


Figure 5.1: Left: $\psi(r)$. Right: $w(r)$ for Huber's loss function.

How do we decide c? Huber suggested $c = 1.345$ and showed it achieved 95% of LSE asymptotically when the true distribution is normal (95% efficiency essentially means the the variance of the betas from OLS is 95% that of the variance of the betas using Huber's loss).

Question 5.3. Since c is fixed, what if our residuals are scaled to very large or small values (e.g. $O(1e5)$ or $O(1e-4)$)? We would have to scale our data beforehand to make it within a sensible range so that $c = 1.345$ makes sense.

Sometimes we prefer the ψ function to "redescend" i.e. $\psi(r) \to 0$ when $|r|$ is large (that is: we fully de-emphasize outliers). Other ψ functions include

Redescending M-estimator (Hampel)

$$\psi(r) = \begin{cases} r & \text{if } 0 \leq |r| \leq a \\ a\text{sign}(r) & \text{if } a \leq |r| \leq b \\ a\frac{c-|r|}{c-b}\text{sign}(r) & \text{if } b \leq |r| \leq c \\ 0 & \text{if } |r| > c \end{cases}$$

The recommended settings are $a = 2, b = 4, c = 8$ (with appropriately scaled data and residuals).

Tukey's biweight

$$\psi(r) = \begin{cases} r\left(1 - \left(\frac{r}{c}\right)^2\right)^2 & \text{if } |r| \leq c \\ 0 & \text{if } |r| > c \end{cases}$$

where $c = 4.685$ is typically used. This is designed to have 95% efficiency as well for a true normal distribution.

6 January 29, 2019

6.1 Remark on robust regression and constants

Remark 6.1. All recommended constants in the various robust regression methods (Huber, Hampel, Tukey) are based on the assumption that $Var(r) = 1$. Therefore in practice we typically need to scale the residuals i.e. $r_i' = \frac{r_i}{s}$ where s is a scale parameter.
One simple solution is to estimate the **median absolute deviation (MAD)**:

$$\text{MAD} = \text{median}(|r_i|)$$

and let $\hat{s} = \frac{\text{MAD}}{0.6745}$. For the standard normal distribution we note that MAD $= 0.6745$.

6.2 Sensitivity curve and breakdown point

Let $T_n(y_1, \ldots, y_n)$ be a population attribute (that is a function of the same points). To see how **sensitive** T_n is to an individual data point, define

$$SC(y) = \frac{T_n(y_1, \ldots, y_{n-1}, y) - T_{n-1}(y_1, \ldots, y_{n-1})}{\frac{1}{n}}$$

which is the difference between $T_n(\cdot)$ (with all n points) and $T_{n-1}(\cdot)$ (with one point y omitted) compare to the contamination size $\frac{1}{n}$.

Example 6.1. Let $T_n(y_1, \ldots, y_n) = \frac{1}{n}\sum_{i=1}^n y_i = \bar{y}_n$ (sample mean).
Note that

$$T_n = \sum_{i=1}^{n-1} y_i + y = \frac{n-1}{n}\bar{y}_{n-1} + y$$

Note that $SC(y)$ is simply

$$\begin{aligned} SC(y) = n(T_n - T_{n-1}) &= (n-1)\bar{y}_{n-1} + y - n\bar{y}_{n-1} \\ &= y - \bar{y}_{n-1} \end{aligned}$$

Definition 6.1 (Breakdown point). *Informally*, the **breakdown point** of a statistic is the largest proportion of contamination before the statistic breaks down.
Formally, let $\vec{z}_i = (x_{i1}, x_{i2}, \ldots, x_{ip}, y_i)^T$ for $i = 1, \ldots, n$ be the ith data vector.
Let $Z = (\vec{z}_1, \ldots, \vec{z}_n)$ be the whole set. Let T be the statistic of interest. The *worst error* for swapping m z_i's is

$$e(m; T, Z) = \sup_{Z_m^*} \|T(Z_m^*) - T(Z)\|$$

where Z_m^* is Z with any of its m data vectors replaced.
The **breakdown point** is then defined as

$$\min\left\{\frac{m}{n} \mid e(m; T, Z) = \infty\right\}$$

Remark 6.2. That is: the breakdown point measures the minimum **proportion** of points required to influence the statistic *significantly*.

Some breakdown point examples:

Sample mean Note we can simply swap out $m = 1$ point arbitrarily such that $e(1; T, Z) \to \infty$ thus the breakdown point is $\frac{1}{n} \to 0$ as $n \to \infty$.

Median The breakdown point is $\frac{1}{2}$ as $n \to \infty$: we need to change at least half of them to aribitrarily influence the median e.g. make it go to infinity.

k% **trimmed mean** The k% trimmed mean is defined as the mean after discarding the lowest k% and highest k% of y_i's.

Breakdown point is k% (we swap out the top $k\% + 1$ points).

6.3 Least median squares (LMS)

Recall for regression, the LSE of $\vec{\beta}$ is

$$\text{argmin}_{\vec{\beta}} \sum_{i=1}^{n} (y_i - \vec{x}_i^T \vec{\beta})^2$$

or equivalently

$$\text{argmin}_{\vec{\beta}} \quad \text{average}(y_i - \vec{x}_i^T \vec{\beta})^2$$

To make it robust for "outliers" or contaminations i.e. to ensure we have a *high breakdown point* we could consider the **least median squares (LMS) estimator**:

$$\vec{\beta}_{LMS} = \text{argmin}_{\vec{\beta}} \quad \text{median}(y_i - \vec{x}_i^T \vec{\beta})^2$$

which has a breakdown point of $\frac{1}{2}$ (compared to a breakdown point of $\frac{1}{n}$ for OLS).

6.4 Least trimmed average sum of squares (LTS)

Similar to how we made our objective function for OLS more robust by considering the median in LMS, we can also consider the **(least) trimmed average sum of squares (LTS) estimator**:

$$\vec{\beta}_{LTS} = \text{argmin}_{\vec{\beta}} \sum_{i=1}^{k} r_{(i)}^2$$

where $r_{(i)}^2$ is the ith smallest squared residual.
Note the breakdown point for LTS is $\frac{n-k+1}{n}$ (compared to a breakdown point of $\frac{1}{n}$ for OLS).

7 January 31, 2019

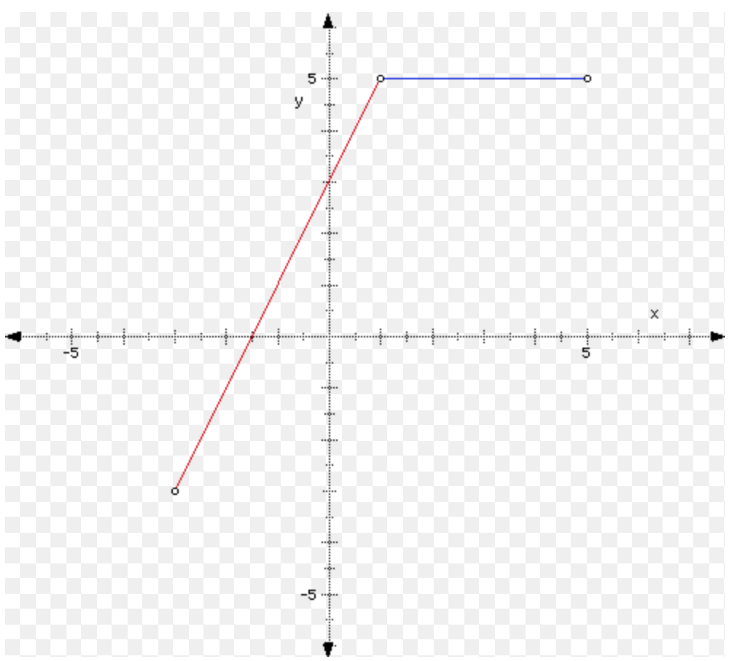
7.1 Local linear regression with k-nearest neighbours

Instead of fitting one linear regression model with all points, we can instead fit local linear regression models for neighbourhoods of points. In essence we are fitting piecewise linear functions.
We first look at **piecewise polynomials** and **splines**.

7.2 Piecewise polynomials (splines)

Definition 7.1 (Spline)**.** We collectively call functions that aim to interpolate and smooth over some distribution **splines**. Piecewise polynomials are a common choice for splines.

For continuous piecewise polynomial functions, the simplest form is **piecewise linear** (as seen before):



which can be specified as a single linear model

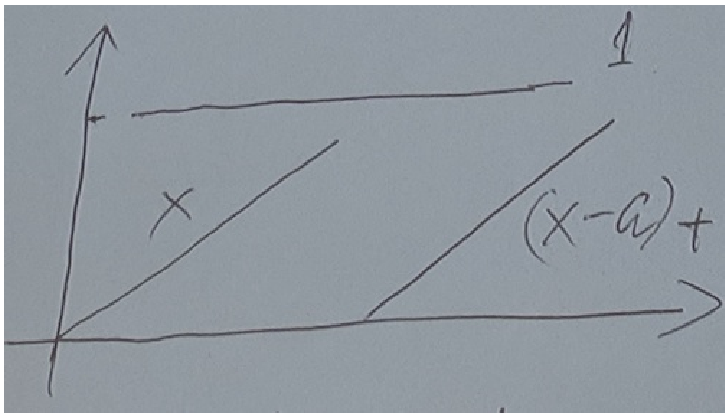
$$f(x) = \beta_0 + \beta_1 x + \beta_2 (x - a) I(x \geq a)$$

Remark 7.1. Piecewise linear is also called the **broken stick** method.

For notation simplicity let us define $(x)_+ = \max(x, 0)$ such that we have

$$f(x) = \beta_0 + \beta_1 x + \beta_2 (x - a)_+$$

Thus our basis functions are $1, x, (x - a)_+$. Here is plot of the basis:



This is an example of the **truncated power series**. We can easily generalize this model to accomodate many break points or knots.
However, picewise linear functions are *not differentiable* at their break points since $f'(x)$ is not continuous.
Recall that for a **piecewise quadratic function** we have

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 (x - a)_+^2$$

where our basis functions are $1, x, x^2, (x - a)_+^2$. Note that a piecewise quadratic model $f(x)$ is indeed differentiable at the break points.

7.3 Cubic splines

Remark 7.2. The most commmonly used spline is the **cubic spline**, which is piecewise cubic where $f(x), f'(x), f''(x)$ are all continuous.

Let $t_1 < t_2 < \ldots, t_k$ be fixed and known knots, where t_1 and t_k are boundary knots and $t_2, \ldots, t_{k-1}$ are interior knots.
Then the basis consists of the functions $1, x, x^2, x^3, (x - t_1)_+^3, \ldots, (x - t_k)_+^3$. That is any **cubic spline** with the above k knots can be expressed as

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \sum_{j=1}^{k} \beta_{j+3} (x - t_j)_+^3$$

Remark 7.3.

1. There are $k + 4$ parameters.
2. $f(x)$ is continuous up to the 2nd derivative.

Proof. This is obviously true between knots. We verify at $x = t_i$:

$$f(t_i) = \beta_0 + \beta_1 t_i + \beta_2 t_i^2 + \beta_3 t_i^3 + \sum_{j=1}^{i-1} \beta_{j+3}(t_i - t_j)_+^3$$

note that $(x - t_j)_+^3 = 0$ for $x < t_{i+1}$ and $j = i+1, \ldots, k$.

Note that $\lim_{x \to t_i^-} f(x) = f(t_i)$ since $(x - t_i)_+ = 0$ if $x < t_i$ so $\lim_{x \to t_i^-} (x - t_i)_+^3 = 0$.

Also $\lim_{x \to t_i^+} f(x) = f(t_i)$ since $\lim_{x \to t_i^+} (x - t_i)_+^3 = 0$.

Therefore $\lim_{x \to t_i^-} f(x) = \lim_{x \to t_i^+} f(x) = f(t_i)$ so f is continuous at t_i for all $i = 1, \ldots, k$.

Similarly we can show this for $f'(x)$ and $f''(x)$. □

8 February 5, 2019

8.1 Natural cubic splines (NCS)

A cubic spline is called a **natural cubic spline** with knots $\{t_1, \ldots, t_k\}$ if $f(x)$ is linear when $x \notin [t_1, t_k]$, that is

$$f(x) = \begin{cases} t_0(x) = a_0 + b_0 x & \text{if } x < t_1 \\ t_k(x) = a_k + b_k x & \text{if } x > t_k \end{cases}$$

Question 8.1. How many free parameters are there in the natural cubic spline?

Answer. Note that in general cubic splines, we have $k + 4$ parameters. If we constrain our spline to be linear at both ends ($x < t_1$ and $x > t_k$) then we essentially remove the quadratic and cubic terms and thus parameters at each end. So we remove 4 parameters and thus we have k free parameters.

To express an NCS, note that for a regular cubic spline we have

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \sum_{j=1}^{k} \beta_{j+3}(x - t_j)_+^3$$

Secondly our constraints are:

$f(x)$ is linear when $x < t_1$ We know that $\beta_4, \ldots, \beta_k + 3$ are already 0 when $x < t_1$.

Thus we need only specify that $\beta_2 = \beta_3 = 0$.

$f(x)$ is linear when $x > t_k$ We require that

$$\sum_{j=1}^{k} \beta_{j+3} = 0$$

$$\sum_{j=1}^{k} \beta_{j+3} t_j = 0$$

since we we expand out the cubic terms

$$\sum_{j=1}^{k} \beta_{j+3}(x^3 - 3t_j x^2 + 3t_j^2 x - t_j^3)$$

we want all the x^3 terms to have 0 coefficients (first term of expansion) and all x^2 terms to also have 0 coefficients (second term of expansion).

These conditions are necessary and sufficient.

Claim. We claim $N_1(x) = 1$, $N_2(x) = x$, and $N_j(x) = d_{j-1}(x) - d_1(x)$ for $j = 3, \ldots, k$ where

$$d_j(x) = \frac{(x - t_j)^3_+ - (x - t_k)^3_+}{t_k - t_j}$$

is a basis for NCS.

Proof. From $\sum_{j=1}^{k} \beta_{j+3} = 0$ we have $\beta_{k+3} = -\sum_{j=1}^{k-1} \beta_{j+3}$.
Thus from the second equation we have

$$\sum_{j=1}^{k-1} \beta_{j+3} t_j + \beta_{k+3} t_k = \sum_{j=1}^{k-1} \beta_{j+3}(t_j - t_k)$$

i.e.

$$\beta_4(t_k - t_1) = -\sum_{j=2}^{k-1} \beta_{j+3}(t_k - t_j)$$

Thus from our original equation (where $\beta_2 = \beta_3 = 0$)

$$\begin{aligned}
f(x) &= \beta_0 + \beta_1 x + \sum_{j=1}^{k} \beta_{j+3}(x - t_j)^3_+ \\
&= \beta_0 + \beta_1 x + \sum_{j=1}^{k-1} \beta_{j+3}(x - t_j)^3_+ + \beta_{k+3}(x - t_k)^3_+ \\
&= \beta_0 + \beta_1 x + \sum_{j=1}^{k-1} \beta_{j+3}\left[(x - t_j)^3_+ - (x - t_k)^3_+\right] \\
&= \beta_0 + \beta_1 x + \beta_4\left[(x - t_1)^3_+ - (x - t_k)^3_+\right] + \sum_{j=2}^{k-1} \beta_{j+3}\left[(x - t_j)^3_+ - (x - t_k)^3_+\right] \\
&= \beta_0 + \beta_1 x + \sum_{j=2}^{k-1} \beta_{j+3}\left[(x - t_j)^3_+ - (x - t_k)^3_+ - \frac{(t_k - t_j)((x - t_1)^3_+ - (x - t_k)^3_+)}{t_k - t_1}\right] \\
&= \beta_0 + \beta_1 x + \sum_{j=2}^{k-1} \beta_{j+3}(t_k - t_j)\left[\frac{(x - t_j)^3_+ - (x - t_k)^3_+}{t_k - t_j} - \frac{(x - t_1)^3_+ - (x - t_k)^3_+}{t_k - t_1}\right] \\
&= \beta_0 + \beta_1 x + \sum_{j=3}^{k} \beta'_{j+2}(d_{j-1}(x) - d_1(x))
\end{aligned}$$

as desired. □

Note that we have 4 separate (linearly independent) constraints on the parameters hence why we lose 4 degrees of freedom.

Let $d_j(x) = \frac{(x-t_j)_+^3 - (x-t_k)_+^3}{t_k - t_j}$, then NCS can be expressed as linear combination of the basis functions

$$\begin{aligned} N_0(x) &= 1 \\ N_1(x) &= x \\ N_j(x) &= (t_k - t_j)\big[d_j(x) - d_1(x)\big] \qquad j = 2, \dots, k-1 \end{aligned}$$

More conveniently we can express the NCS as

$$f(x) = \sum_{j=1}^{k} \beta_j N_j(x)$$

where $N_1(x) = 1, N_2(x) = x$ and $N_j(x) = d_{j-1}(x) - d_1(x)$ for $j = 3, \dots, k$.

Remark 8.1. 1. If $x < t_1$, then $d_j(x) = 0 \Rightarrow N_j(x) = 0$ for $j = 3, \dots, k$.

2. If $x > t_k$, then $d_j(x) = \frac{(x-t_j)^3 - (x-t_k)^3}{t_k - t_j}$ reduces to a quadratic function of x where the coefficient of x^2 term is 3.

   Since $N_j(x) = d_{j-1}(x) - d_1(x)$ then it is a linear function of x if $x > t_k$ for $j = 3, \dots, k$.

Definition 8.1 (Regression splines)**.** The fixed-knot splines, such as cubic splines and NCS, are called **regression splines**.

8.2 Fitting NCS

Let $y_i = f(x_i) + \epsilon_i$ for some response y_i and explanatory variates x_i and some arbitrary continuous function $f(\cdot)$. We can approximate/regress $f(x)$ by $\sum_{j=1}^{k} \beta_j N_j(x)$ (NCS) i.e.

$$y_i \approx \sum_{j=1}^{k} \beta_j N_j(x_i) + \epsilon_i$$

Now we simply fit the following linear model with design matrix

$$X = \begin{bmatrix} N_1(x_1) & \dots & N_k(x_1) \\ \vdots & \ddots & \vdots \\ N_1(x_n) & \dots & N_k(x_n) \end{bmatrix}$$

where

$$\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}$$
$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Remark 8.2. The problem becomes a regular regression problem with design matrix generated from the basis functions N_j's.

8.3 General function fitting with basis funcitons

We extend our method for fitting NCS: more generally for a p-dimensional input vector $\vec{x}$, we can consider the following appproximation to $f(\vec{x})$

$$f(\vec{x}) = \sum_{j=1}^{k} \beta_j h_j(\vec{x})$$

where $\{h_j\}$ are a series of basis functions.

That is: we approximate $f(\vec{x})$ as a linear basis expansion. Then we form the design matrix $X = \begin{bmatrix} h_j(\vec{x}_i) \end{bmatrix}$ where i indexes the row (ith sample) and j indexes the column (jth basis function).
Some examples:

1. $h_j(\vec{x}) = x_j$ for $j = 1, \dots, p$ is the original linear model where basis functions are the jth component
2. $h_j(\vec{x}) = \log(x_j)$ are arbitrary transformations
3. $h_j(\vec{x}) = x_j^k$ for $k \in \mathbb{N}$ is polynomial regression
4. $h_j(\vec{x}) = N_j(\vec{x})$ is NCS

9 February 7, 2019

9.1 Choosing k for NCS

Recall that the basis functions for NCS are

$$\begin{aligned} N_0(x) &= 1 \\ N_1(x) &= x \\ N_i(x) &= d_{j-1}(x) - d_1(x) \qquad j = 3, \dots, k \end{aligned}$$

We still need to choose a k and our knots $t_1, \dots, t_k$.
Some examples of how to choose k and knots:

Equal-distance knots We choose k first arbitrarily e.g. $k = 5$, then we use an equal-distance grid between the min and max of x_i's.

Quantiles **Quantiles** are also a popular choice e.g. $\frac{i}{k-1}$ quantiles for each x_i, $i = 0, \dots, k-1$.

Degrees of freedom Alternatively we can instead specify the degrees of freedom for an NCS i.e. the number of free parameters. For $df = k$, we would have $k - 2$ knots (if intercept term is also included). Usually knots are placed at equal distance quantiles.

9.2 Smoothing splines

Consider the following **penalized** regression problem

$$\hat{f}_\lambda(x) = \operatorname{argmin}_f \sum_{i=1}^{n} [y_i - f(x_i)]^2 + \lambda \int_{-\infty}^{\infty} [f''(x)]^2 \, \mathrm{d}x$$

Remark 9.1. 1. $\sum_{i=1}^{n} [y_i - f(x_i)]^2$ is the sum of squared residuals which measures the goodness of fit.

2. $\int_{-\infty}^{\infty}[f''(x)]^2\,\mathrm{d}x$ measures the "roughness" of $f(x)$.

   Remark 9.2. Note that we try to minimize the integral over the $f''(x)$ (squared), which is essentially minimizing $f''(x)$ so that it is close to 0.

   For example, if $f(x) = \beta_0 + \beta_1 x$ (OLS) then $f''(x) = 0$ thus $\int_{-\infty}^{\infty}[f''(x)]^2 = 0$ i.e. no penalty for OLS.

3. The role of λ: if $\lambda = 0$ then we have no roughness penalty and we will minimize the SSR over all functions and $\hat{f}_\lambda(x)$ is the interpolating line.

   If $\lambda = \infty$ then we will force $\int_{-\infty}^{\infty}[f''(x)]^2\,\mathrm{d}x = 0$ thus $\hat{f}_\lambda(x)$ is the ordinary least square fit.

4. Remarkably we can show that $\hat{f}_\lambda(x)$ is just the natural cubic spline with knots at distinct values of $\{x_i\}_{i=1}^n$.

5. NCS is the "smoothest" interpolator.

   For any complex function $f(x)$ if we only know the value of k points $\{f(t_i)\}_{i=1}^k$ then we can use $\{t_i, f(t_i)\}_{i=1}^k$ to determine an NCS $s(x)$ such that $s(t_i) = f(t_i)$ for $i = 1, \ldots, k$.

   Claim.
   $$\int_{-\infty}^{\infty}[s''(x)]^2\,\mathrm{d}x \leq \int_{-\infty}^{\infty}[f''(x)]^2\,\mathrm{d}x$$

   Proof. Left as exercise in assignment. □

Definition 9.1 (Smoothing spline)**.** We call the function fitted by the penalized regression a **smoothing spline**.

We determine the β for the NCS smoothing spline. Note that

$$\hat{f}_\lambda(x) = \sum_{j=1}^{k}\beta_j N_j(x)$$

that is

$$\hat{\beta}_\lambda = \mathrm{argmin}_f \sum_{i=1}^{n}[y_i - \sum_{j=1}^{k}\beta_j N_j(x)]^2 + \lambda\int_{-\infty}^{\infty}[\sum_{j=1}^{k}\beta_j N_j''(x)]^2\,\mathrm{d}x$$

Note that we can re-express this in matrix notation where

$$\sum_{i=1}^{n}[y_i - \sum_{j=1}^{k}\beta_j N_j(x)]^2 = (Y - X\vec{\beta})^T(Y - X\vec{\beta})$$

where

$$X = \begin{bmatrix} N_1(x_1) & \ldots & N_k(x_1) \\ \vdots & \ddots & \vdots \\ N_1(x_n) & \ldots & N_k(x_n) \end{bmatrix}$$

Also

$$\begin{aligned}
\int_{-\infty}^{\infty}[\sum_{j=1}^{k}\beta_j N_j''(x)]^2\,\mathrm{d}x &= \int_{-\infty}^{\infty}[\sum_{j=1}^{k}\beta_j N_j''(x)][\sum_{l=1}^{k}\beta_l N_l''(x)]\,\mathrm{d}x \\
&= \int_{-\infty}^{\infty}[\sum_{j=1}^{k}\sum_{l=1}^{k}\beta_j\beta_l N_j''(x)N_l''(x)]\,\mathrm{d}x \\
&= \sum_{j=1}^{k}\sum_{l=1}^{k}\beta_j\beta_l\big(\int_{-\infty}^{\infty}N_j''(x)N_l''(x)\,\mathrm{d}x\big) \\
&= \vec{\beta}^T N\vec{\beta}
\end{aligned}$$

where $N = (N_{jl}) = \int_{-\infty}^{\infty} N_j''(x)N_l''(x)\,\mathrm{d}x$ (i, j-th entry is N_{jl}).
Therefore we can let

$$\begin{aligned}
S(\vec{\beta}) &= (Y - X\vec{\beta})^T(Y - X\vec{\beta}) + \lambda\vec{\beta}^T N\vec{\beta} \\
&= Y^TY - \vec{\beta}^TX^TY - Y^TX\vec{\beta} + \vec{\beta}^TX^TX\vec{\beta} + \lambda\vec{\beta}^TN\vec{\beta} \\
&= Y^TY - 2Y^TX\vec{\beta} + \vec{\beta}^T(X^TX + \lambda N)\vec{\beta}
\end{aligned}$$

and $\hat{\vec{\beta}}_\lambda = \mathrm{argmin} S(\vec{\beta})$.
Recall that for matrix Y, A and vector $\vec{c}$

$$\begin{aligned}
\frac{\partial \vec{c}^T Y}{\partial Y} &= \vec{c}^T \\
\frac{\partial Y^T A Y}{\partial Y} &= 2Y^T A^T
\end{aligned}$$

thus we have

$$\begin{aligned}
&\frac{\partial S(\vec{\beta})}{\partial \vec{\beta}} = 0 = -2Y^TX + 2\vec{\beta}^T(X^TX + \lambda N)^T \\
&\Rightarrow (X^TX + \lambda N)\hat{\vec{\beta}}_\lambda = X^TY \\
&\Rightarrow \hat{\vec{\beta}}_\lambda = (X^TX + \lambda N)^{-1}X^TY
\end{aligned}$$

To calculate the *effective number of parameters* or effective df (edf): recall for NCS we have k knots and in OLS with $X_{n\times p}$

$$\hat{Y} = HY = X(X^TX)^{-1}X^TY$$

where the number of parameters is $df = tr(H)$.
Now in the smoothing spline, we have

$$\hat{Y}_\lambda = X(X^TX + \lambda N)^{-1}X^TY = A_\lambda Y$$

where the effective number of parameters is $df_\lambda = tr(A_\lambda)$.

Remark 9.3. When $\lambda \to \infty$, $df_\lambda \to 2$.

10 February 14, 2019

10.1 B-splines

A comptutationally efficient alternative to cubic splines and NCS is the **B-spline**.
The basis functions of B-splines are strictly local. For a degree d B-spline (e.g. $d = 3$ for a cubic B-spline), each basis function is non-zero over the interval of $d+2$ adjacent knots (and zero everywhere else) i.e. $d+1$ intervals.
Advantages of B-spline:

1. Numerically stable: recall cubic splines have x^3 terms which grows fast as $x \to \infty$. B-splines are fitted to a restriction of x (the $d+5$ knots).

2. Computationally efficient: when # of knots k is large. More specifically least squares estimation with n observations and k variables takes $O(nk^2 + k^3)$ operations. If $k \to n$ then this becomes $O(n^3)$. B-splines reduces this cost to $O(n)$ (since $k \in O(d)$) becomes constant.

We define the 0-degree B-spline basis:

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } t_i \leq x < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

where $B_{i,0}(x)$ is the interval indicator function. It is also known as the **Haar basis function**.
In general for a d-degree B-spline, we define its basis as:

$$B_{i,d}(x) = \frac{x - t_i}{t_{i+d} - t_i} B_{i,d-1}(x) + \frac{t_{i+d+1} - x}{t_{i+d+1} - t_{i+1}} B_{i+1,d+1}(x)$$

After we compute the basis functions given our x, we can fit the model as an OLS or robust LR model. In `R`, we can use the function `bs` in the package `splines` to generate the *B-spline basis functions* (note there are no intercepts included). This will give us a design matrix with $d+k$ basis functions (so $d+k$ degrees of freedom) where d is the degree and k is the number of knots (d starts at 0 for the constant function).
Then we simply feed this to `lm` or `rlm` as usual (which will subsequently introduce the bias term). Note that `lm` will add one more degree of freedom with the intercept for $(d+1)+k$ degrees of freedom.
Similarly we can generate NCS basis functions with `ns` in `splines`.

10.2 Smoothing splines in R

To fit smoothing splines (penalized splines), we can use the `smooth.spline` function.
First specify an appropriate degrees of freedom `df`.
Let nx be the number of distinct values of x. We use nx knots if $nx \leq 49$ and $O(nx^{0.2})$ knots if $nx > 49$.
Although it is not strictly a smoothing spline when $nx > 49$ it is very close to one.

Remark 10.1. Since smoothing splines are penalized for their "smoothness" this allows us to choose a high number of knots.

11 February 26, 2019

11.1 KNN local linear regression

An alternative to fitting splines with prespecified knots is to fit the data more locally by considering a neighbourhood at each point x.

We take the k nearest neighbours for every x and compute the mean response value of the neighbours. This then becomes the fitted value at x_i and we may *linearly interpolate* or even *quadratically interpolate* (or even higher order polynomial interpolation) between points.
This can be accomplished in `R` with `knn.reg` from the `FNN` package.

Remark 11.1. As the neighbourhood size (k) increases, the smoother the function becomes.

Instead of taking the mean response based on the k neighbours, we can instead use the value from any fitted model based on those k neighbours (e.g. `lm`, `rlm` with Huber, Tukey's, etc., `ltsreg`).

Remark 11.2. We can think of KNN local linear regression as weighted linear regression where $w_j = 0$ if x_j is outside the neighbourhood of x_i.

11.2 Kernel local linear regression

In the KNN case, we essentially assign equal weighting amongst all k neighbours. Instead we can use some kernel to assign higher weights to points closer to x_i and lower weight to points farther away from x_i.
A kernel $K(t)$ must satisfy the following

$$\int K(t)\,\mathrm{d}t = 1 \qquad \int tK(t)\,\mathrm{d}t = 0 \qquad \int t^2K(t)\,\mathrm{d}t < \infty$$

where the first two standardize $K(t)$ and the last constraint ensures weights are spread along the real line but not too much weight are in the extremes.
Some examples of kernels:

Epanechinikov $K(t) = \frac{3}{4}(1-t^2)I(|t| \leq 1)$

Tukey's tri-cube $K(t) = (1-|t|^3)^3 I(|t| \leq 1)$

Gaussian $K(t) = \frac{1}{\sqrt{2\pi e^{-t^2/2}}}$

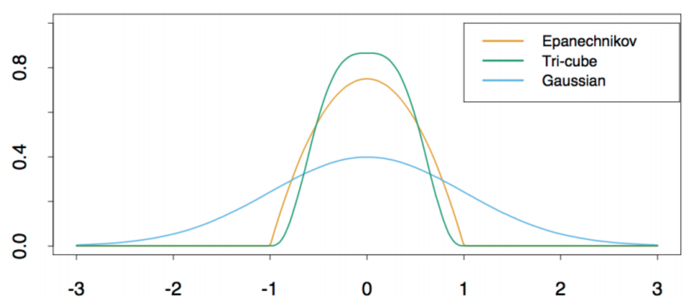


Figure 11.1: Graph of kernels used in kernel local linear regression.

Thus for a bandwidth parameter h the weight function for neighbour x_i for current point x is defined as:

$$w_i = w(x, x_i) = \frac{K\left(\frac{x_i - x}{h}\right)}{\sum_{j=1}^{N} K\left(\frac{x_j - x}{h}\right)}$$

For the mean response $\hat{\mu}(x)$ we take the weighted average or the **Nadaraya-Watson estimator**:

$$\hat{\mu}(x) = \sum_{i=1}^{N} w_i y_i$$

Remark 11.3. The **boundary effect** occurs when no points lie on one side of the kernel and thus the weights are distributed in a biased way to the points on the other side. This occurs at the extremes of the explanatory variate space.

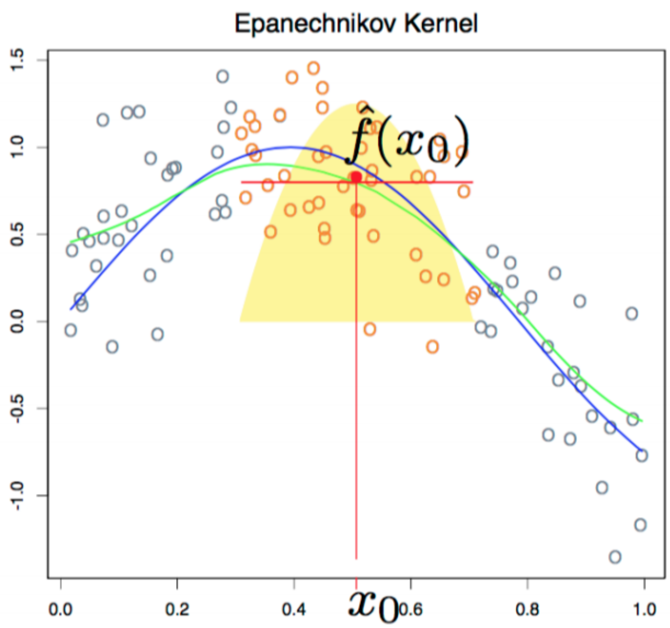


Figure 11.2: The boundary effect causes the kernel fitted line (green) at the left end to bias the fitted value higher since most the available points are on the right of the kernel and have a higher response value.

To avoid the boundary effect, local regression is typically used.
Local linear is simply and good at boundaries and local quadratic is good at interior points.
Higher order regression is rarely used.

In `R` we can use the `loess` function where `span` defines the proportion of points in the local neighbourhood. The kernel used is **Tukey's tri-cube**.

11.3 Linear smoother

A smoothing spline with for fixed λ is an example of a **linear smoother** where

$$\begin{aligned}\hat{Y}_\lambda &= X(X^TX + \lambda N)^{-1}X^TY \\ &= S_\lambda Y\end{aligned}$$

A **linear smoother** is a linear combination of the y_i's with S_λ being the **smoother matrix**.
Consider a regression with a small number p of basis functions. That is for basis functions $b_1, \ldots, b_p$ (e.g. the NCS basis) let the matrix

$$[B_{n\times p}]_{i,j} = b_j(x_i)$$

That is: the i,j-th element is $b_j(x_i)$.
Then the ordinary least squares (OLS) solution for the fitted values are

$$\hat{Y} = B(B^T B)^{-1} B^T Y$$

where $H_B = B(B^T B)^{-1} B^T$ (the hat matrix).

Claim. It can be shown (for any hat matrix) that the column space of B satisfies $C(B) = C(H_B)$. Note however B has p columns whereas H_B has n columns, so there is some redundancy.

Proof. Since H_B is symmetric we take its eigendecomposition

$$\begin{aligned}
H_B &= UPU^T \\
&= \begin{bmatrix} \vec{u}_1 & \dots & \vec{u}_n \end{bmatrix} \begin{bmatrix} \rho_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \rho_n \end{bmatrix} \begin{bmatrix} \vec{u}_1 \\ \vdots \\ \vec{u}_n \end{bmatrix} \\
&= \sum_{j=1}^{n} \rho_i \vec{u}_i \vec{u}_i^T
\end{aligned}$$

where $\rho_1 \geq \ldots \geq \rho_n \geq 0$ are the eigenvalues and $\vec{u}_1, \ldots, \vec{u}_n$ are the corresponding orthonormal eigenvectors. Then

$$\begin{aligned}
\hat{Y} &= H_B Y \\
&= \sum_{i=1}^{n} \rho_i \vec{u}_i \vec{u}_i^T Y \\
&= \sum_{i=1}^{n} \rho_i \langle \vec{u}_i, Y \rangle \vec{u}_i && \langle \cdot, \cdot \rangle \text{ inner product}
\end{aligned}$$

Thus Y is first projected to the orthonormal basis $\{\vec{u}_1, \ldots, \vec{u}_n\}$ then modulated by $\{\rho_1, \ldots, \rho_n\}$.
Because H_B is idempotent and assuming B is full rank (i.e. $\text{rank}(B) = p$) then $P^n = P$ for all $n \in \mathbb{N}$ thus

$$\rho_i = \begin{cases} 1 & \text{if } i = 1, \ldots, p \\ 0 & \text{if } i = p+1, \ldots, n \end{cases}$$

□

12 February 28, 2019

12.1 Reinsh form of smoother matrix

For a smoothing spline and assuming the n x_i values are distinct, then

$$\left[X_{n \times n} \right]_{i,j} = h_j(x_i)$$

where h_j is the jth NCS basis function.

Claim. Given the smooth spline

$$\hat{Y}_\lambda = X(X^T X + \lambda N)^{-1} X^T Y = S_\lambda Y$$

we claim the smoother matrix S_λ in a linear smoother can be written in the **Reinsch form**

$$S_\lambda = (I + \lambda K)^{-1}$$

where $K = (X^T)^{-1} N X^{-1}$ does not depend on λ and N is the **roughness penalty matrix** for NCS, that is $N = \big(N_{jl}\big)_{n\times n}$ and

$$N_{jl} = \int_{-\infty}^{\infty} N_j''(x) N_l''(x)\,\mathrm{d}x$$

Proof. First remark that X is a square matrix since we assume all x_i values are distinct so

$$\begin{aligned}
S_\lambda &= X(X^TX + \lambda N)^{-1}X^T \\
&= \big[(X^T)^{-1}(X^TX + \lambda N)X^{-1}\big]^{-1} \\
&= \big(I + \lambda (X^T)^{-1} N X^{-1}\big)^{-1} \\
&= (I + \lambda K)^{-1}
\end{aligned}$$

where $K = (X^T)^{-1} N X^{-1}$. □

12.2 Penalty form of linear smoother

It can be shown that $\hat{Y}_\lambda = S_\lambda Y$ is the solution to the minimization problem

$$\min \quad (Y - \vec{\mu})^T (Y - \vec{\mu}) + \lambda \vec{\mu}^T K \vec{\mu}$$

where K is known as the **penalty matrix** where K is symmetric and has eigendecomposition

$$K = VDV^T$$

with $D = \operatorname{diag}(d_1, \dots, d_n)$ the eigenvalues where $d_i \geq 0$ and $V = (\vec{v}_1, \dots, \vec{v}_n)$ is an orthonomal matrix (of eigenvectors).

Remark 12.1.

1. $K = \sum_{i=1}^n d_i \vec{v}_i \vec{v}_i^T$ (sum of matrices) and $\vec{\mu}^T K \vec{\mu} = \sum_{i=1}^n d_i \langle \vec{v}_i, \vec{\mu}\rangle^2$.
   This implies that $\vec{\mu}$ is penalized more in the directions of v_i's with large d_i values.
2. It can be shown that $d_{n-1} = d_n = 0$.

It is straightforward to show that

$$S_\lambda = V(I + \lambda D)^{-1} V^T$$

is the eigendecomposition of S_λ with eigenvalues

$$\rho_i(\lambda) = \frac{1}{1 + \lambda d_{n-i+1}} \qquad i = 1, \dots, n$$

Remark 12.2.

1. S_λ and K share the same eigenvectors which *do not depend on* λ.
2. Large eigenvalues d_i of D leads to small eigenvalues $\rho_i(\lambda)$ of S_λ.
3. Large λ leads to small eigenvalues $\rho_i(\lambda)$.
4. $\rho_1(\lambda) = \rho_2(\lambda) = 1$ since $d_{n-1} = d_n = 0$.
   All other $\rho_i(\lambda)$'s are less than 1.

Returning to our original fitted value $\vec{\mu}$

$$\begin{aligned}\vec{\mu} &= S_\lambda Y \\ &= \sum_{i=1}^n \rho_i(\lambda)\vec{v}_i\vec{v}_i^T Y \\ &= \sum_{i=1}^n \rho_i(\lambda)\langle \vec{v}_i, Y\rangle \vec{v}_i\end{aligned}$$

that is: we project Y down to the every eigenvector $\vec{v}_i$ and scale each projection by corresponding eigenvalue $\rho_i(\lambda)$. Note the first two vectors are not shrunk by $\rho_i(\lambda)$ but the rest are shrunk towards 0 since $\rho_i(\lambda) < 1$.

12.3 Regression vs smoothing splines

How do smoothing splines compare to regresson splines (e.g. OLS)?
To draw a parallel with OLS, recall

$$\hat{Y} = \sum_{i=1}^n \rho_i \langle \vec{u}_i, Y\rangle \vec{u}_i$$

where

$$\rho_i = \begin{cases} 1 & \text{if } i = 1, \ldots, p \\ 0 & \text{if } i = p+1, \ldots, n \end{cases}$$

Comparing the two, OLS selects the eigenvector basis where eigenvalues are 1 and drops the other eigenvectors (**hard thresholding**) whereas smoothing splines shirnk Y in the direction of eigenvectors according to their corresponding eigenvalues $\rho_i(\lambda)$.
For this reason OLS or regression splines are called **projection smoothers** and smoothing splines are **shrinking smoothers**.

Remark 12.3. 1. The sequence of $\vec{v}_i$, ordered by decreasing eigenvalues $\rho_i(\lambda)$, appear to increase in complexity (i.e. roughness or "wiggleness").

2. Recall the effective degrees of freedom is

$$\text{df}_\lambda = \text{tr}(S_\lambda) = \sum_{i=1}^n \frac{1}{1+\lambda d_i}$$

thus if we want a specific df_λ we can simply linear search for the corresponding λ (since the d_i's are also fixed).

13 March 5, 2019

13.1 Local linear regression as a linear smoother

We show that local linear regression is indeed a linear smoother.
For target value x (point we are doing local regression about), local linear regression is equivalent to solving the weighted optimization problem

$$\text{argmin}_{\alpha,\beta} \sum_{i=1}^n k_h(x - x_i)(y_i - (\alpha + \beta x_i))^2$$

and our fitted value at x from local regression is $\hat{f}(x) = \alpha(x) + \beta(x)x$ (k_h is our kernel function used in local linear regression).
Note the above optimization problem has an explicit solution. Let

$$B = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$
$$W(x) = \text{diag}(k_h(x - x_1), \ldots, k_h(x - x_n))$$

where B is $n \times 2$ and W is $n \times n$.
Then $\hat{f}(X)$ ($n \times 1$ vector of fitted values) can be re-rewritten as

$$\begin{aligned} \hat{f}(X) &= (1, x)\big(B^T W(X) B\big)^{-1} B^T W(X) Y \\ &= l^T(X) Y \end{aligned}$$

where $l^T(X) = (1, x)\big(B^T W(X) B\big)^{-1} B^T W(X)$.

Remark 13.1. 1. Local linear regression is a linear smoother i.e. $\hat{f}(x)$ is a linear combination of y_i's.

2. Recall in smoothing splines
$$\hat{Y}_\lambda = S_\lambda Y$$
and we define $df_\lambda = \text{tr}(S_\lambda)$.
If we let
$$L_h = \begin{bmatrix} l^T(x_1) \\ l^T(x_2) \\ \vdots \\ l^T(x_n) \end{bmatrix}$$
where L_h is $n \times n$, then $\hat{Y}_h = L_h Y$ and we efine $df_h = tr(L_h)$.

14 March 7, 2019

14.1 Multivariate local regression

We want to make a prediction at target value $\vec{x} = (x_1, \ldots, x_p)^T$. A simple kernel function we could use in local regression in $\mathbb{R}^p$

$$K_h(\vec{x}) = \frac{1}{h} K\big(\frac{\|\vec{x}\|}{h}\big)$$

where $\|\cdot\|$ is the Euclidean norm.

Remark 14.1. The issue with the Euclidean norm is we weight every coordinate/variate x_i equally. If a variate has less importance we should not give it the same weight as other variates in the kernel.
That is: the kernel we have above is equally-skewed about all variates x_i and gives equal weight to each coordinate.

We use **structured local regression** instead. This is a more general approach where we use a positive semidefinite matrix $A_{p \times p}$ to weight each coordinate, that is:

$$K_{h,A}(\vec{x}) = \frac{1}{h} K\big(\frac{\vec{x}^T A \vec{x}}{h}\big)$$

Typically A is chosen to be symmetric and there are $\frac{p(p+1)}{2}$ elements in A.
A popular choice is a diagonal matrix i.e. $A = \text{diag}(a_1, \dots, a_p)$ where a_i modulates the variance of the kernel along the ith dimension (as $a_i \to \infty$ then the larger and hence more smoothed out the kernel is along the ith dimension).

14.2 Multivariate regression splines with tensor products

An extension of regression splines to $\mathbb{R}^p$ are **tensor products**.

Example 14.1. Let $p = 2$ and $\vec{x} = (x_1, x_2)^T \in \mathbb{R}^2$.
Let $\{h_{11}, h_{12}, \dots, h_{1k_1}\}$ a set of spline basis functions on x_1, and $\{h_{21}, \dots, h_{2k_2}\}$ a set of spline basis functions on x_2.
Consider the following tensor product basis with $k_1 \times k_2$ functions where

$$g_{jk}(\vec{x}) = h_{1j}(x_1)h_{2k}(x_2)$$

and

$$f(\vec{x}) = \sum_{j,k} \beta_{jk} g_{jk}(x)$$

that is f is a *multiplicative/interaction-based model.*

We note that the number of parameters with a tensor product basis grows **exponentially** with p.

14.3 Multivariate smoothing splines with thin plates

An extension of smoothing splines to $\mathbb{R}^p$ are with **thin plate splines**.

Example 14.2. Let $p = 2$. We solve for

$$\hat{f}_\lambda = \text{argmin}_f \{ \sum_{i=1}^{n} [y_i - f(\vec{x}_i)^2]^2 + \lambda J(f) \}$$

where

$$J(f) = \int_{\mathbb{R}} \int_{\mathbb{R}} \left(\frac{\partial^2 f}{\partial x_1^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 f}{\partial x_2^2} \right)^2 \mathrm{d}x_1 \, \mathrm{d}x_2$$

We can show the optimal solution has the form (at a given target $\vec{x}$)

$$f_\lambda(\vec{x}) = \beta_{0,\lambda} + \beta_\lambda^T \vec{x} + \sum_{i=1}^{n} \alpha_i h_i(\vec{x})$$

where the generator function is the **radial basis function** $h_i(\vec{x}) = \|\vec{x} - \vec{x}_i\|^2 \log\|\vec{x} - \vec{x}_i\|$.
Note $\vec{x}_i$ for $i = 1, \dots, n$ are our control/knot points.

14.4 Curse of dimensionality

Any dataset with in a large dimension $\mathbb{R}^p$ is **sparse**.

Example 14.3. Consider fixed-width neighbourhoods in e.g. local regression.
For $p = 1$, assume data points are uniformly distributed across the domain $x \in [0, 1]$. Suppose for a target x we have a neighbourhood $x \pm 0.05$. Then we will capture $\approx 10\%$ of points.

For $p = 2$ again assume data points are uniformly distributed across $[0, 1] \times [0, 1]$. Suppose we have a neighbourhood $x \pm 0.1$: note that we capture an area of $0.2 \times 0.2 = 0.04$ which only captures $\approx 4\%$ of points!
As p increases, even as we increase the width of our neighbourhood along each dimension linearly our data becomes so sparse our neighbourhood becomes relatively small.

14.5 Structured regression additive approach

For predictor values $\vec{x} = (x_1, \ldots, x_p)^T$, instead of modelling

$$\mu(\vec{x}) = f(x_1, \ldots, x_p)$$

which may be some arbitrary possibly interactive function of every variate we consider the **additive model**

$$\mu(\vec{x}) = \alpha + \sum_{j=1}^{p} f_j(x_j)$$

where f_j's can be linear functions or any smooth function of x_j.

Remark 14.2. We can extend the above model to allow a *limited number of interactions.*
For example if p is small we can consider additional pairwise interactions

$$f(\vec{x}) = \alpha + \sum_{j=1}^{p} f_j(x_j) + \sum_{j=1}^{p} \sum_{k=1}^{p} f_{jk}(x_j, x_k)$$

We may also be more selective, e.g. for $p = 5$

$$f(\vec{x}) = \alpha + f_1(x_1) + f_2(x_2, x_3) + f_3(x_4, x_5)$$

15 March 12, 2019

Review of assignment 3 solutions and slides on generative additive model.

16 March 14, 2019

Review of assignment 4 solutions and more slides on generative additive model.

17 March 19, 2019

17.1 Tuning parameter selection

Suppose we are given the true model $y_i = f(x_i) + \epsilon_i$ where $E(\epsilon_i) = 0$ and $Var(\epsilon_i) = \sigma^2$, observations $(x_i, y_i)_{n=1}^{n}$ (training set T), and prediction model $\hat{f}(x)$.
The methods considered so far typically involve some tuning parameter e.g. the # of knots in regression spline, λ in smoothing spline, bandwidth h in local regression, etc.
These tuning parameters are considered "complexity parameters" that regulate the complexity (i.e. degrees of freedom) of the prediction model.
In prediction we want $\hat{f}$ to provide good estimates for **future observations**.

Definition 17.1 (Test error)**.** We define the **test error** as:

$$Err_T = \mathbb{E}\big((Y - \hat{f}(X))^2 \mid T\big)$$

where the expectation is over the *true joint distribution* of (X, Y) i.e. the population.

Definition 17.2 (Expected test error)**.** We define the **expected test error** as:

$$Err = \mathbb{E}\big((Y - \hat{f}(X))^2\big) = \mathbb{E}(Err_T)$$

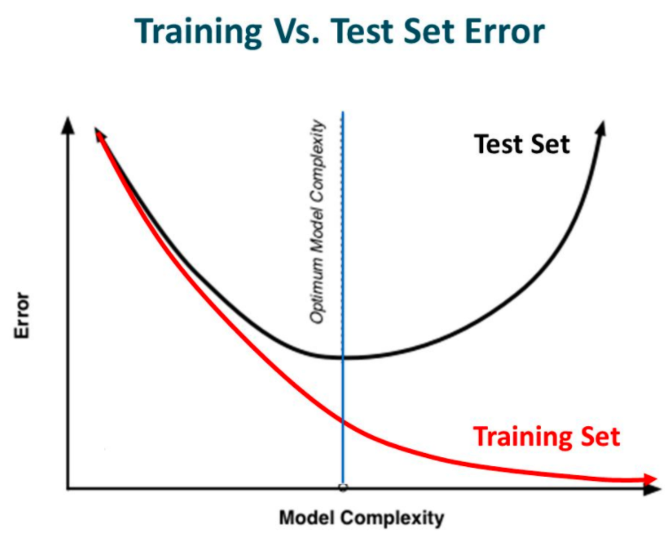
where the distribution is over the distribution of (X, Y) and the random generation of training sets.

Definition 17.3 (Training error)**.** We define the **training error** as:

$$\overline{Err} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 = \frac{RSS}{n}$$

However $\overline{Err}$ uses the same data twice (once for producing $\hat{f}$ and once for calculating the error) and does not track Err well.

We note that as the model complexity grows, training error decreases while our test error will increase after the optimal complexity:



Test error increases after a certain point due to **overfitting** to the training set.
To **regularize** our model for complexity, some solutions include:

Information criteria let d denote the number of parameters. We define the **Akaike Information Criterion (AIC)** as:

$$AIC = 2d + n\log(RSS)$$

and the **Bayesian Information Criterion (BIC)**, which has a larger regularization effect, as:

$$BIC = \log(n)d + n\log(RSS)$$

Remark 17.1. The information criteria are only useful for comparing models for the same training sample. Their absolute values are meaningless.

Cross-validation (CV) Recall we can estimate the test error Err by repeatedly sampling test sets from the population.

We hold out a part of the training set as our "population" (cross-validation set) and construct our model on the remaining training set. We can then validate our complexity and model on the cross-validation set as an estimate of the test error.

This error on the cross-validation set is called the **cross-validation error**.

k-fold CV

1. Given a training set T, randomly partition it into k disjoint equal-sized parts ("folds") $T_1, \ldots, T_k$.
2. For every $i = 1, \ldots, k$, we train our model on partitions $T_1, \ldots, T_{i-1}, T_{i+1}, T_k$ to obtain $\hat{f}^{(k)}$, then we evaluate $\hat{f}^{(k)}$ on T_i to get the cross-validation error for each T_i. Let $i(j)$ be the fold T_i corresponding to example j, then the overall cross-validation error is:

$$CV(\hat{f}) = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{f}^{(i(j))}(x_j))^2$$

The choice of $k = n$ is called the **leave-one-out (LOO)** CV. The justification for LOO CV: Let $\hat{f}^{-i}(x_i)$ be the fitted value of x_i without using (x_i, y_i) during training. Then

$$\begin{aligned}
\mathbb{E}(y_i - \hat{f}^{-i}(x_i))^2 &= \mathbb{E}(y_i - f(x_i) + f(x_i) - \hat{f}^{-i}(x_i))^2 \\
&= \mathbb{E}(y_i - f(x_i))^2 + 2\epsilon_i\mathbb{E}(f(x_i) - \hat{f}^{-i}(x_i)) + \mathbb{E}(f(x_i) - \hat{f}^{-i}(x_i))^2 \\
&= \sigma^2 + \mathbb{E}(f(x_i) - \hat{f}^{-i}(x_i))^2 \\
&\approx \sigma^2 + \mathbb{E}(f(x_i) - \hat{f}^{-i}(x_i))^2
\end{aligned}$$

that is LOO CV provdies an approximate estimate of the test error Err (up to a constant σ^2).

Thus minimizing the LOO CV error is essentially minimizing the test error.

Remark 17.2. Since LOO CV requires fitting the data n times, for large n then this is infeasible.

Remark 17.3. For most **linear smoothers** where $\hat{Y} = SY$ where S is the smoother matrix it can be shown that

$$CV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}^{-i}(x_i))^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}(x_i)}{1 - s_{ii}}\right)^2$$

where s_{ii} is the ith diagonal element of S. We can thus simply fit the data once and weight the squared residuals by $\frac{1}{(1-s_{ii})^2}$.

The above proof for OLS is in A2 Q2 part (d).

Generalized Cross Validation (GCV) For any **linear smoother** $\hat{Y} = SY$ we define the GCV error as

$$GCV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}(x_i)}{1 - \frac{tr(S)}{n}}\right)^2$$

where we use the average trace $tr(S)/n$ instead of each individual s_{ii}.

Note that LOO CV is approximately unbiased for Err, but can have high variance due to the n training sets being very similar to one another. On the other hand, a small k tends to have large bias but small variance. To balance bias and variance, $k = 5$ or $k = 10$ is recommended.

18 March 26, 2019

18.1 Tree-based methods

A decision tree is essentially local linear regression with K neighbourhoods, where at some iteration with i neighbourhoods we partition some neighbourhood to get $i+1$ neighbourhoods. We aim to optimize the objective function:

$$\sum_{i=1}^{N}(y_i - \sum_{k=1}^{K} I_{R_k}(x_i)\hat{\mu}_k)^2 + \lambda K$$

where $I_{R_k}(x_i) = 1$ if $x_i \in R_k$ neighbourhood k and $\hat{\mu}_k$ is the average of the points in the neighbourhood k. This is equivalent to local average regression.
Optimizing the above is still computationally difficult (since there are a combinatorial number of different $K \in \mathbb{N}$ neighbourhoods to consider for N points.