Machine Learning

A. Supervised Learning: Linear Models & Fundamentals

A.4. High-Dimensional Data

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Syllabus

Fri. 26.10.	(1)	0. Introduction
		A. Supervised Learning: Linear Models & Fundamentals
Fri. 2.11.	(2)	A.1 Linear Regression
Fri. 9.11.	(3)	A.2 Linear Classification
Fri. 16.11.	(4)	A.3 Regularization
Fri. 23.11.	(5)	A.4 High-dimensional Data
		B. Supervised Learning: Nonlinear Models
Fri. 30.11.	(6)	B.1 Nearest-Neighbor Models
Fri. 7.12.	(7)	B.4 Support Vector Machines
Fri. 14.12.	(8)	B.3 Decision Trees
Fri. 21.12.	(9)	B.5 A First Look at Bayesian and Markov Networks
		— *Christmas Break* —
Fri. 11.1.	(10)	B.2 Neural Networks
		C. Unsupervised Learning
Fri. 18.1.	(11)	C.1 Clustering
Fri. 25.1.	(12)	C.2 Dimensionality Reduction
Fri. 1.2.	(13)	C.3 Frequent Pattern Mining
Fri. 8.2.	(14)	Q&A

Outline

1. Variable Interactions and Polynomial Models
2. Parameter Variance
3. Variable Selection via Forward and Backward Search
4. Minimizing a Function via Coordinate Descent
5. L1 Regularization / The Lasso

High-Dimensional Data

High-dimensional data occurs in different situations:

1. Data that comes naturally with many predictors.
   - e.g., text classification
     (# predictors = # words in the bag-of-words representation, e.g., 30.000)

2. Models that extract many predictor variables from objects to classify.
   - variable interactions
   - derived variables
   - complex objects such as graphs, texts, etc.
     - Situation 1 often really is a special case of this one.

3. Data with few examples compared to the number of variables ("small n, large p").
   - gene expression / microarray data

Outline

Need for higher orders

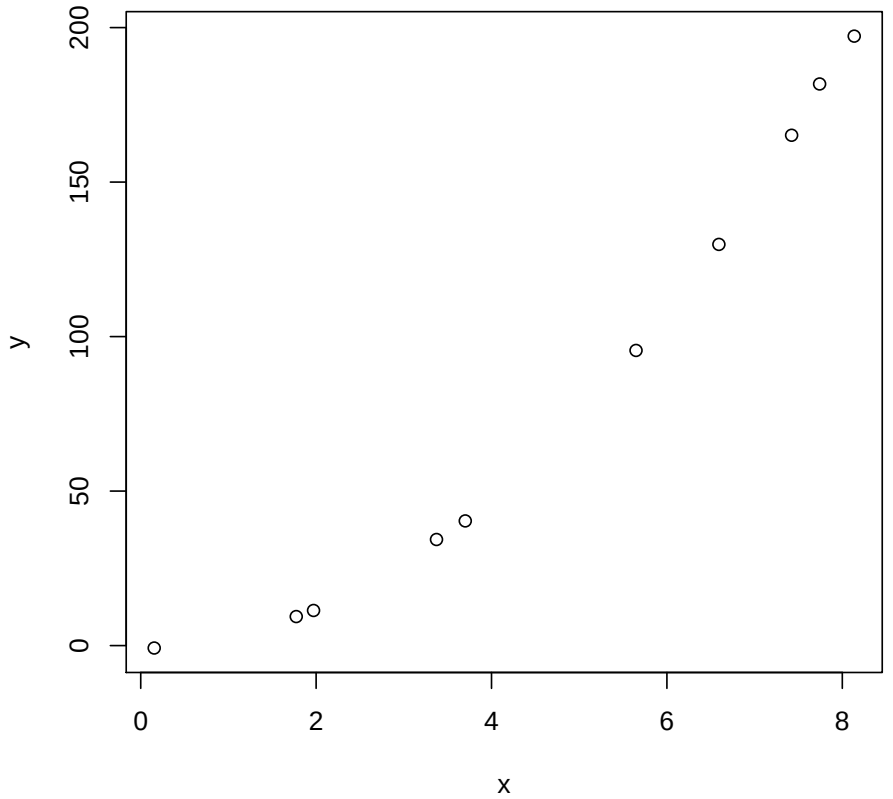
Assume a target variable does not depend linearly on a predictor variable, but say quadratic.

Example: way length vs. duration of a moving object with constant acceleration a.

$$s(t) = \frac{1}{2}at^2 + \epsilon$$

Can we catch such a dependency?

Can we catch it with a linear model?



Need for general transformations

To describe many phenomena, even more complex functions of the input variables are needed.

Example: the number of cells n vs. duration of growth t:

$$n = \beta e^{\alpha t} + \epsilon$$

n does not depend on t directly, but on $e^{\alpha t}$ (with a known α).

Need for variable interactions

In a linear model with two predictors

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

Y depends on both, X_1 and X_2.

But changes in X_1 will affect Y the same way, regardless of X_2.

There are problems where X_2 mediates or influences the way X_1 affects Y, e.g. : the way length s of a moving object vs. its constant velocity v and duration t:

$$s = vt + \epsilon$$

Then an additional $1s$ duration will increase the way length not in a uniform way (regardless of the velocity), but a little for small velocities and a lot for large velocities.

v and t are said to **interact**: y does not depend only on each predictor separately, but also on their product.

Derived variables

All these cases can be handled by looking at **derived variables**, i.e., instead of

$$Y = \beta_0 + \beta_1 X_1^2 + \epsilon$$
$$Y = \beta_0 + \beta_1 e^{\alpha X_1} + \epsilon$$
$$Y = \beta_0 + \beta_1 X_1 \cdot X_2 + \epsilon$$

one looks at

$$Y = \beta_0 + \beta_1 X_1' + \epsilon$$

with

$$X_1' := X_1^2$$
$$X_1' := e^{\alpha X_1}$$
$$X_1' := X_1 \cdot X_2$$

Derived variables are computed before the fitting process and taken into account either additional to the original variables or instead of.

Polynomial Models

Polynomial models of degree d take into account systematically all interactions of d different variables (including powers up to degree d):

$$\hat{y}(x) := \hat{\theta}_0 + \sum_{m=1}^{M} \hat{\theta}_m x_m \qquad \text{degree 1}$$

Polynomial Models

Polynomial models of degree d take into account systematically all interactions of d different variables (including powers up to degree d):

$$\hat{y}(x) := \hat{\theta}_0 + \sum_{m=1}^{M} \hat{\theta}_m x_m \qquad \text{degree 1}$$

$$\hat{y}(x) := \hat{\theta}_0 + \sum_{m=1}^{M} \hat{\theta}_m x_m + \sum_{m=1}^{M} \sum_{l=m}^{M} \hat{\theta}_{m,l} x_m x_l \qquad \text{degree 2}$$

Polynomial Models

Polynomial models of degree d take into account systematically all interactions of d different variables (including powers up to degree d):

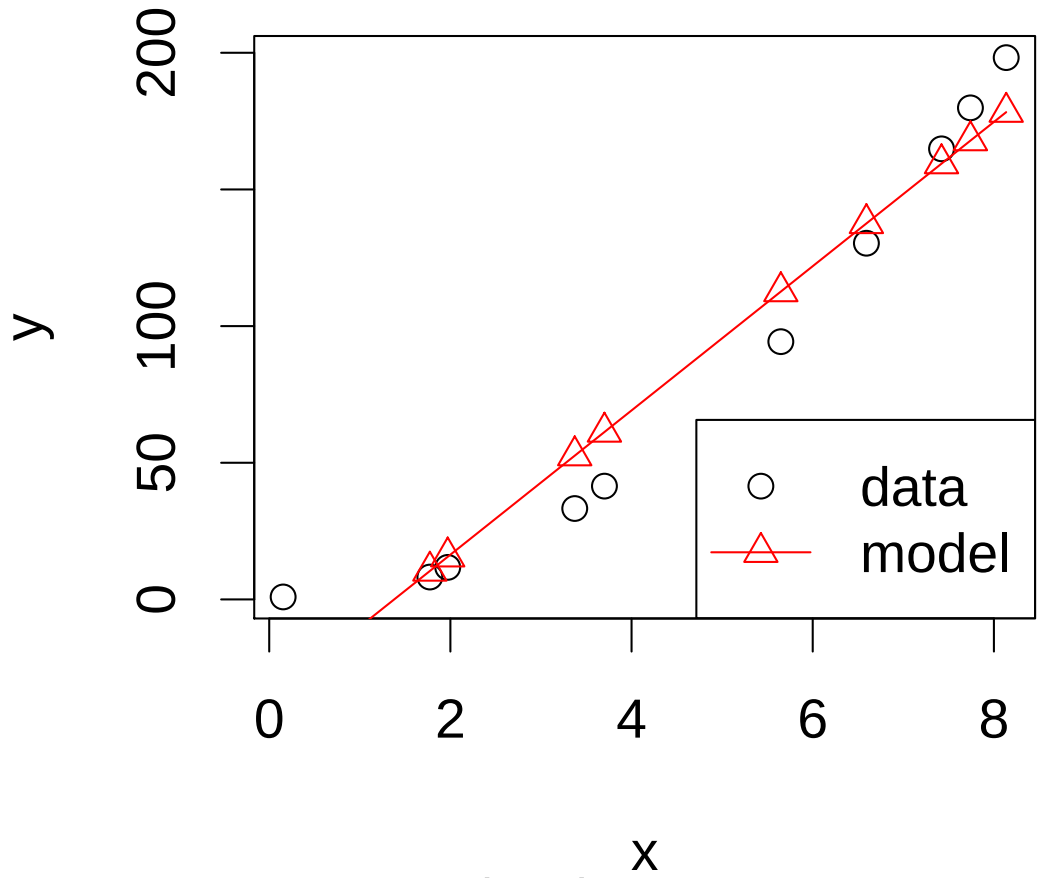
$$\hat{y}(x) := \hat{\theta}_0 + \sum_{m=1}^{M} \hat{\theta}_m x_m \qquad \text{degree 1}$$

$$\hat{y}(x) := \hat{\theta}_0 + \sum_{m=1}^{M} \hat{\theta}_m x_m + \sum_{m=1}^{M} \sum_{l=m}^{M} \hat{\theta}_{m,l} x_m x_l \qquad \text{degree 2}$$

$$\hat{y}(x) := \hat{\theta}_0 + \sum_{m=1}^{M} \hat{\theta}_m x_m + \sum_{m=1}^{M} \sum_{l=m}^{M} \hat{\theta}_{m,l} x_m x_l + \cdots$$

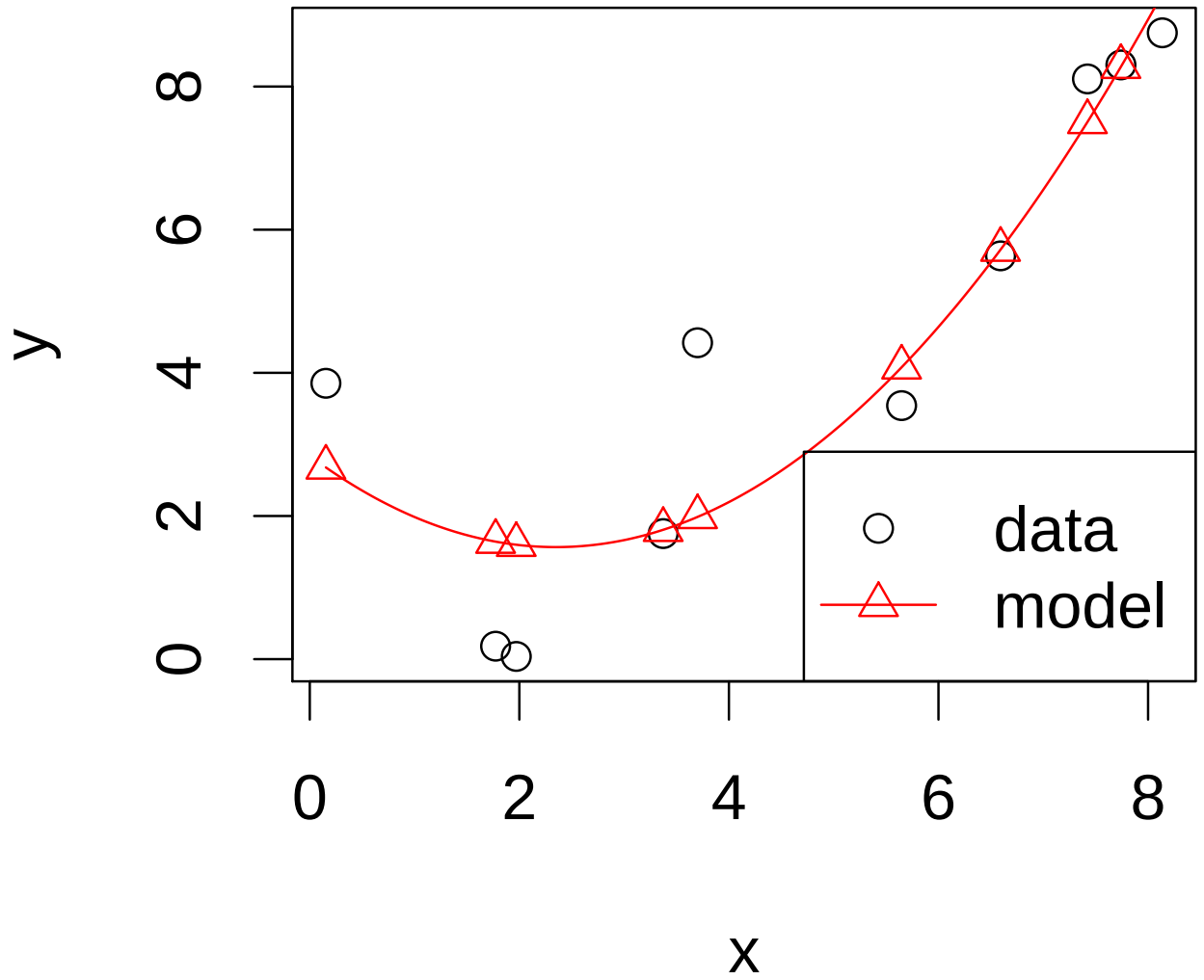
$$+ \sum_{m_1=1}^{M} \sum_{m_2=m_1}^{M} \cdots \sum_{m_d=m_{d-1}}^{M} \hat{\theta}_{m_1,m_2,\ldots,m_d} x_{m_1} x_{m_2} \cdots x_{m_d} \qquad \text{degree } d$$

High Polynomial Degress, High Model Complexity

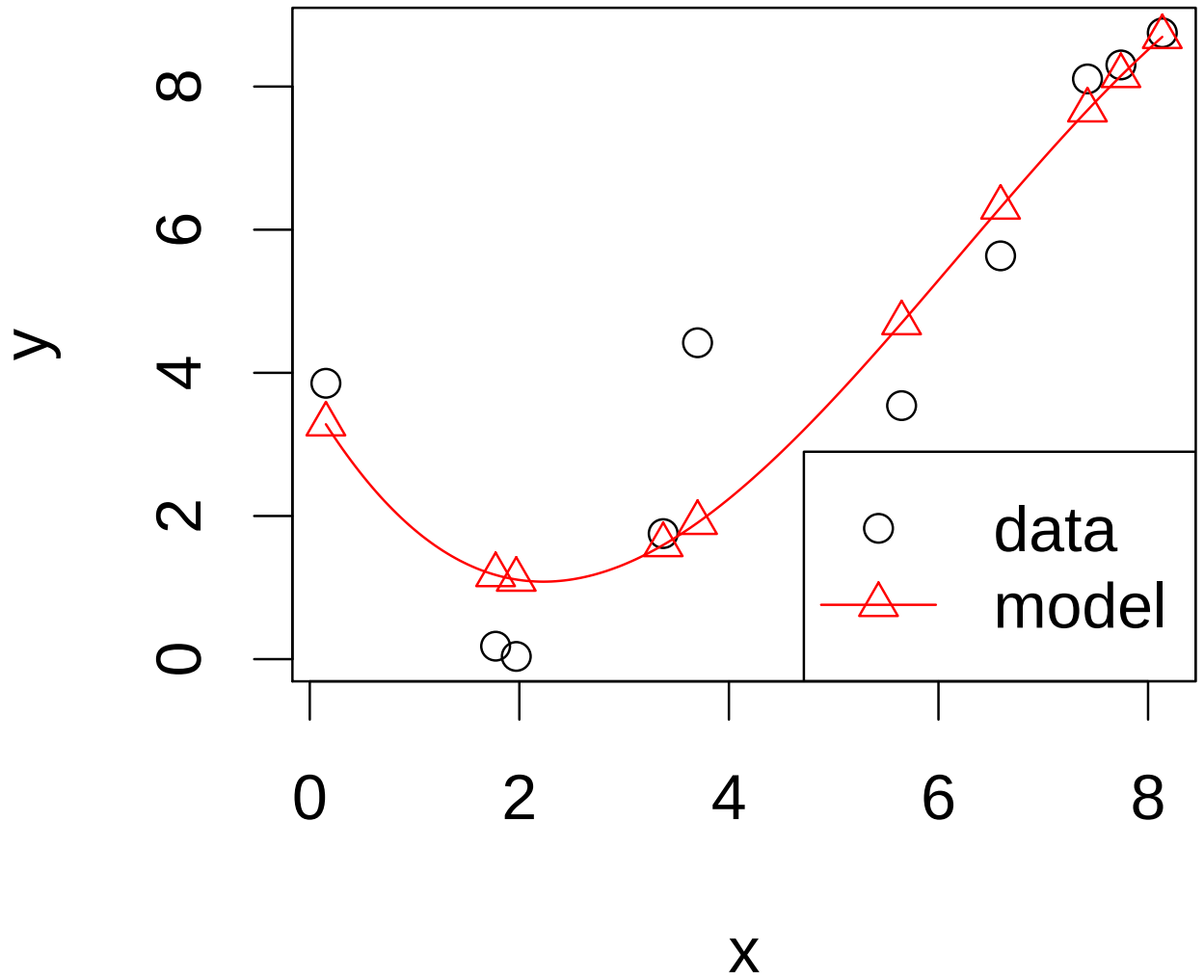


If a model does not well explain the data,
e.g., if the true model is quadratic, but we try to fit a linear model,
one says, the model **underfits**.

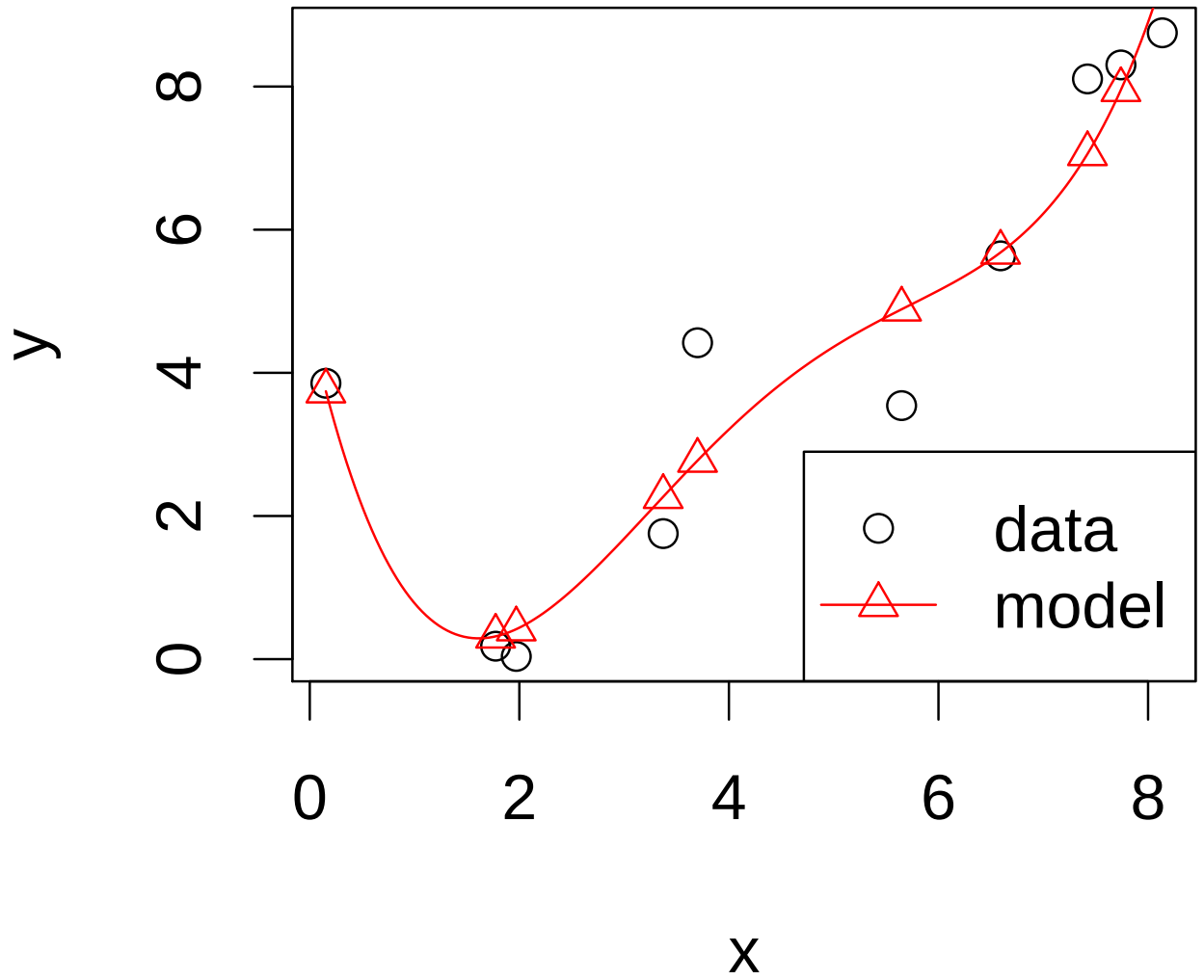
High Polynomial Degress, High Model Complexity



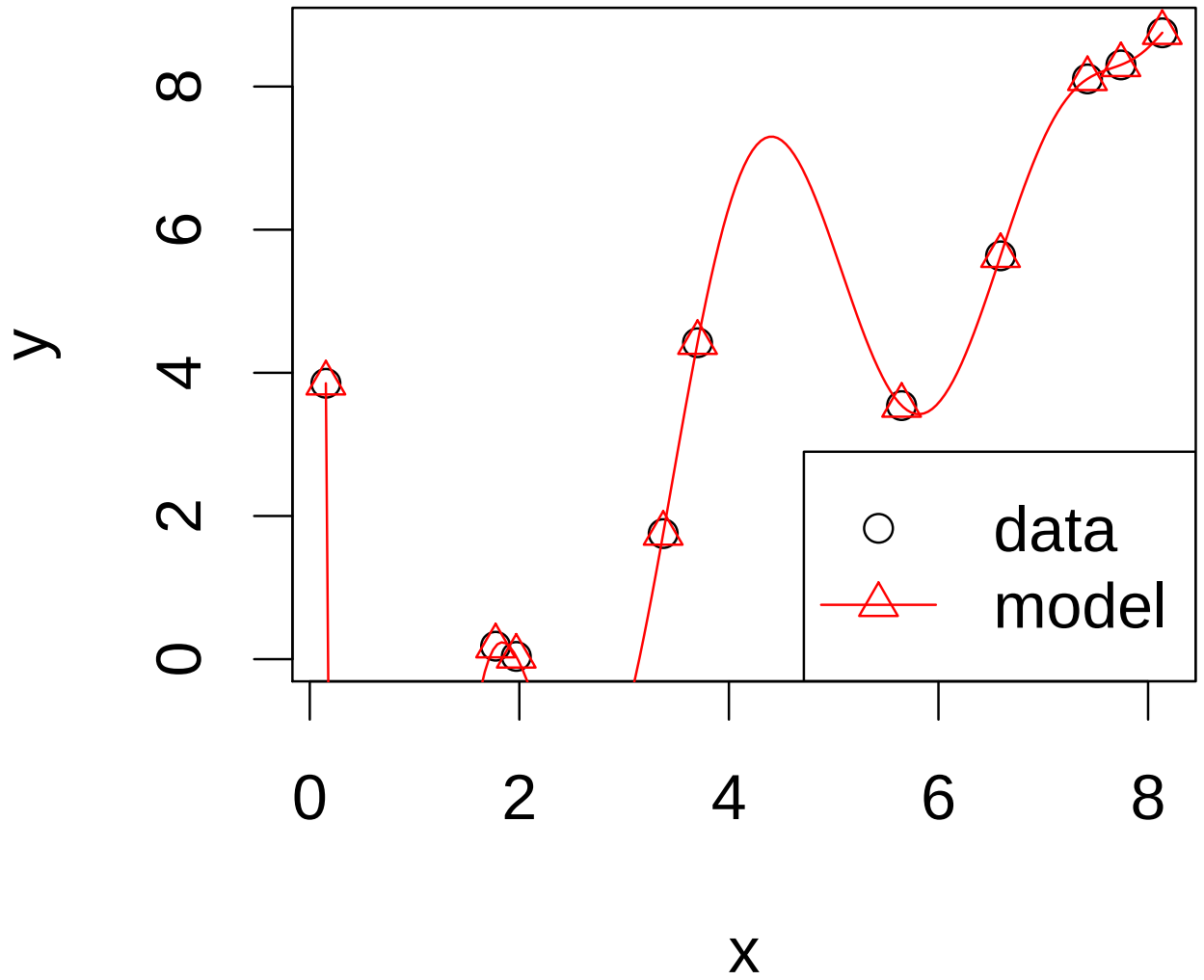
High Polynomial Degress, High Model Complexity



High Polynomial Degress, High Model Complexity



High Polynomial Degress, High Model Complexity



High Polynomial Degress, High Model Complexity

If to data

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

consisting of N points we fit

$$\begin{aligned} X &= \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_{N-1} X^{N-1} \\ &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_{N-1} X_{N-1}, \quad X_i := X^i \end{aligned}$$

i.e., a polynomial with degree $N - 1$,
then this results in an **interpolation** of the data points
(if there are no repeated measurements, i.e., points with the same X.)

As the polynomial

$$\hat{y}(X) = \sum_{n=1}^{N} y_n \prod_{m \neq n} \frac{X - x_m}{x_n - x_m}$$

is of this type, and has minimal RSS $= 0$.

Variable Types and Coding

The most common variable types:

numerical / interval-scaled / quantitative

- differences and quotients etc. are meaningful,
- usually with domain $\mathcal{X} := \mathbb{R}$,
- e.g., temperature, size, weight.

nominal / discrete / categorical / qualitative / factor

- differences and quotients are **not** defined,
- usually with a finite, enumerated domain,
- e.g., $\mathcal{X} := \{\text{red}, \text{green}, \text{blue}\}$
  or $\mathcal{X} := \{\text{a}, \text{b}, \text{c}, \ldots, \text{y}, \text{z}\}$.

ordinal / ordered categorical

- levels are ordered, but differences and quotients are not defined,
- usually with a finite, enumerated domain,

Variable Types and Coding

Nominals are usually encoded as a set of binary **dummy variables** (aka **indicator variables**, **one hot encoding**):

$$\delta_{x_0}(X) := \begin{cases} 1, & \text{if } X = x_0, \\ 0, & \text{else} \end{cases}$$

one for each $x_0 \in \mathcal{X}$ (but one).

Example: $\mathcal{X} := \{\text{red}, \text{green}, \text{blue}\}$

one variable X with 3 levels: red, green, blue

$\downarrow$ replace by

two variables $\delta_{\text{red}}(X)$ and $\delta_{\text{green}}(X)$ with 2 levels each: 0, 1

X	$\delta_{\text{red}}(X)$	$\delta_{\text{green}}(X)$
red	1	0
green	0	1
blue	0	0
—	1	1

Outline

The Normal Distribution (also Gaussian)

written as:

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

with parameters:

- μ mean,
- σ standard deviance.

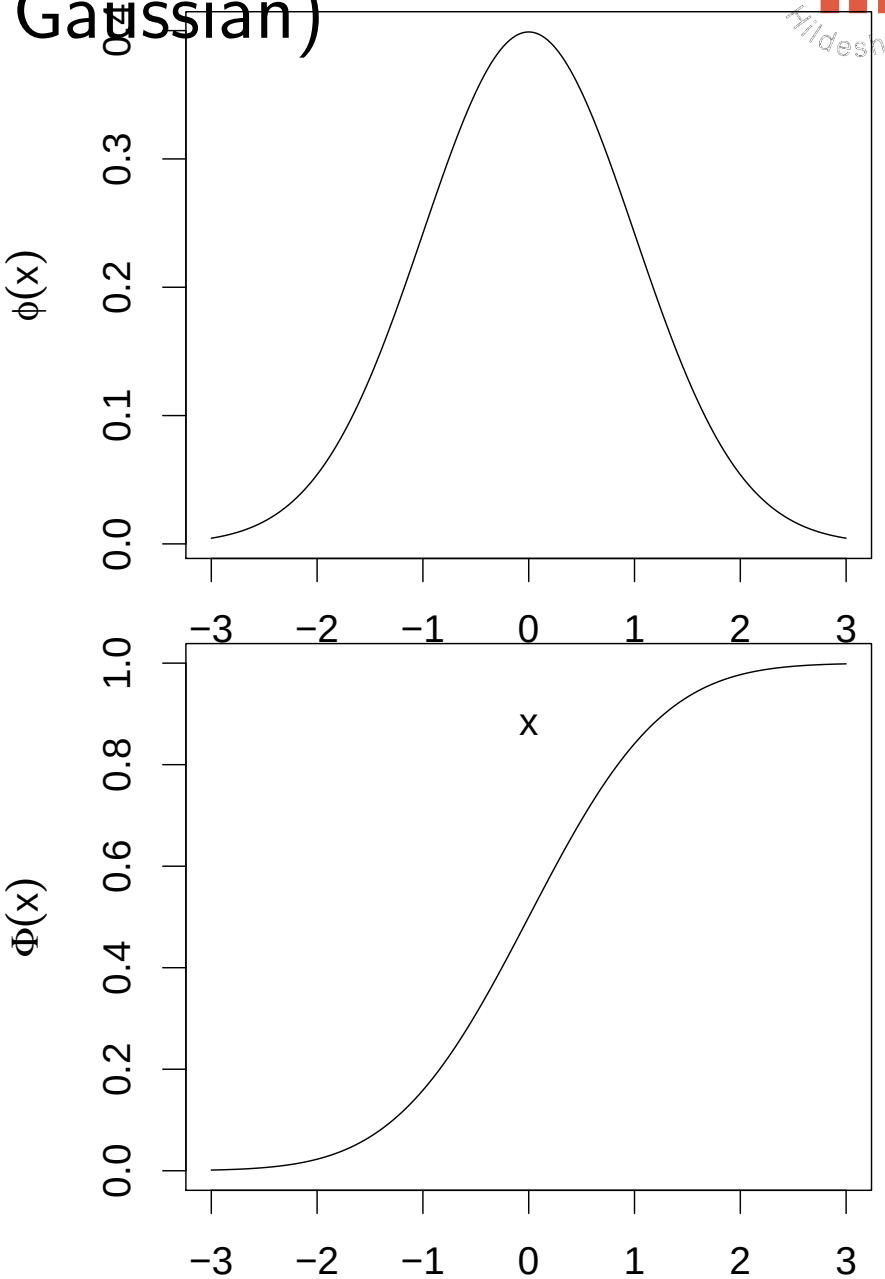
probability density function (pdf):

$$\phi(x) := \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

cumulative distribution function (cdf):

$$\Phi(x) := \int_{-\infty}^{x} \phi(t)dt$$

Φ^{-1} is called **quantile function**.



The t Distribution

written as:

$$X \sim t_p$$

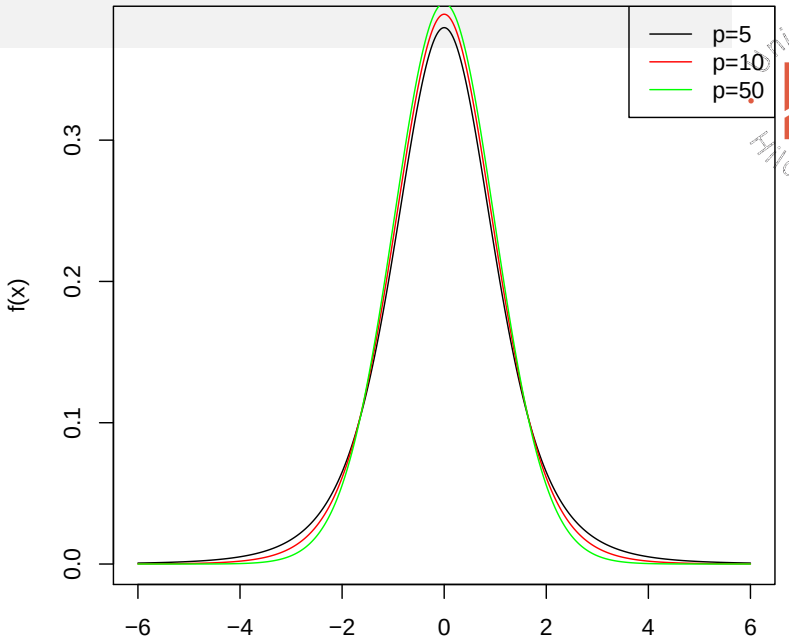
with parameter:

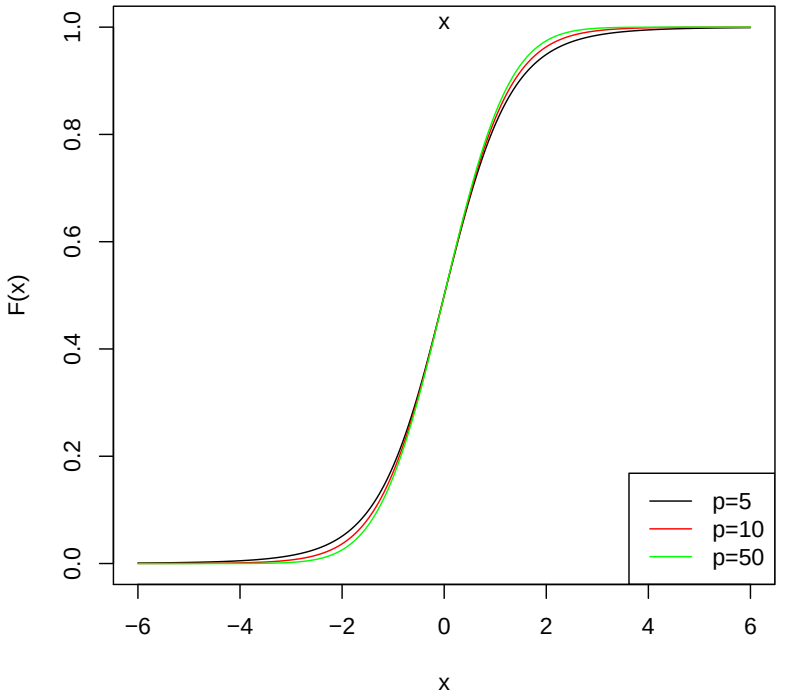
p degrees of freedom.

probability density function (pdf):

$$p(x) := \frac{\Gamma(\frac{p+1}{2})}{\sqrt{p\,\pi}\,\Gamma(\frac{p}{2})}(1 + \frac{x^2}{p})^{-\frac{p+1}{2}}$$

$$t_p \overset{p\to\infty}{\longrightarrow} \mathcal{N}(0,1)$$





The χ^2 Distribution

written as:

$$X \sim \chi^2_p$$

with parameter:

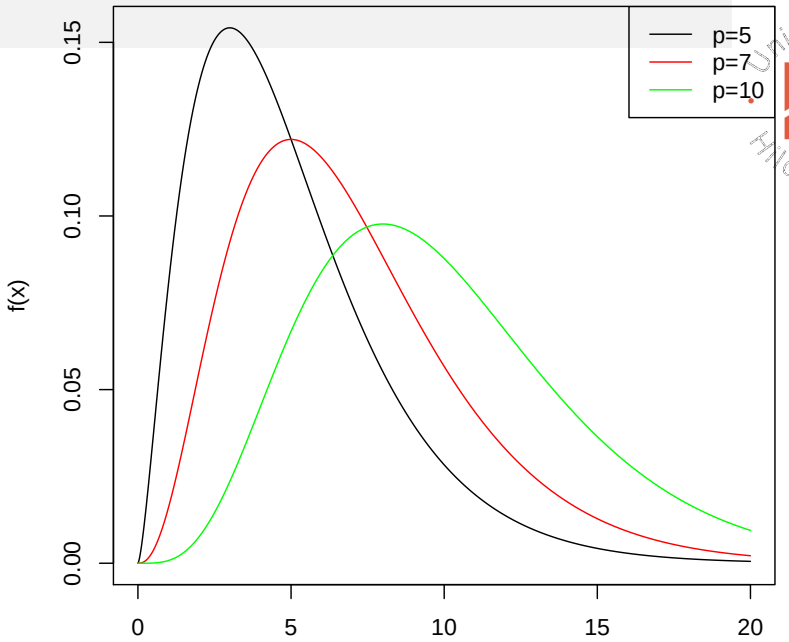
p degrees of freedom.

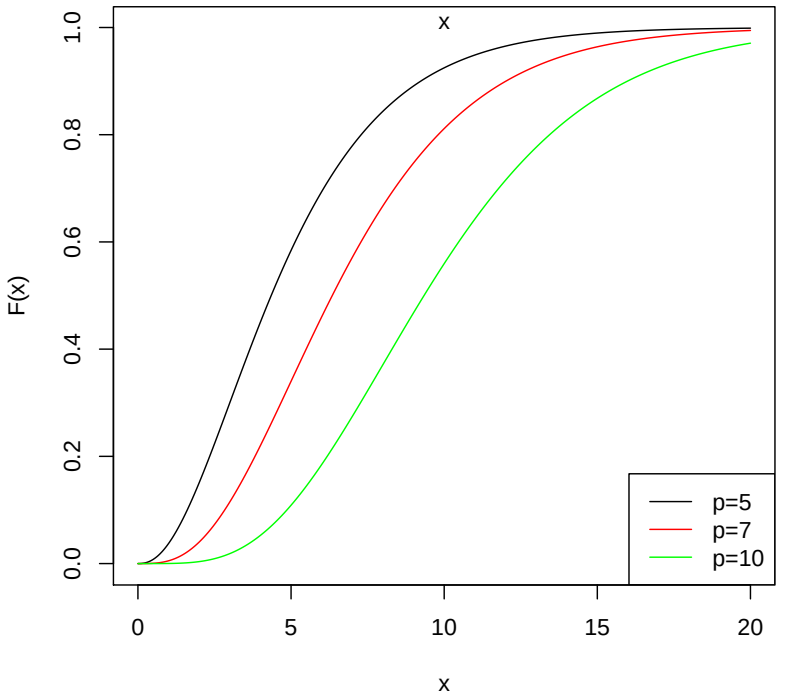
probability density function (pdf):

$$p(x) := \frac{1}{\Gamma(p/2)2^{p/2}} x^{\frac{p}{2}-1} e^{-\frac{x}{2}}, \quad x \geq 0$$

If $X_1, \ldots, X_p \sim \mathcal{N}(0,1)$, then

$$Y := \sum_{i=1}^{p} X_i^2 \sim \chi^2_p$$





Parameter Variance for Linear Regression

$\hat{\beta} = (X^T X)^{-1} X^T y$ is an unbiased estimator for β (i.e., $\mathbb{E}(\hat{\beta}) = \beta$).
Its variance is

$$\mathbb{V}(\hat{\beta}) = (X^T X)^{-1} \sigma^2$$

proof: assume ground truth $Y = X\beta + \epsilon$, $\mathbb{E}(\epsilon) = 0$, $\mathbb{V}(\epsilon) = \sigma^2 I$:

$$\hat{\beta} = (X^T X)^{-1} X^T y = (X^T X)^{-1} X^T (X\beta + \epsilon) = \beta + (X^T X)^{-1} X^T \epsilon$$

$$\rightsquigarrow \quad \mathbb{E}(\hat{\beta}) = \beta + \mathbb{E}(\epsilon) = \beta$$

$$\begin{aligned}
\mathbb{V}(\hat{\beta}) &= \mathbb{E}((\hat{\beta} - \mathbb{E}(\hat{\beta}))(\hat{\beta} - \mathbb{E}(\hat{\beta}))^T) \\
&= \mathbb{E}((X^T X)^{-1} X^T \epsilon \epsilon^T X (X^T X)^{-1}) \\
&= (X^T X)^{-1} X^T \mathbb{E}(\epsilon \epsilon^T) X (X^T X)^{-1} \\
&= (X^T X)^{-1} \sigma^2
\end{aligned}$$

Parameter Variance for Linear Regression

An unbiased estimator for σ^2 is

$$\hat{\sigma}^2 = \frac{1}{N-M}\sum_{n=1}^{N}\hat{\epsilon}_n^2 = \frac{1}{N-M}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2$$

For Gaussian errors $\epsilon \sim \mathcal{N}(0, \sigma^2)$:

$$\hat{\beta} \sim \mathcal{N}(\beta, (X^T X)^{-1}\sigma^2)$$

and

$$(N-M)\hat{\sigma}^2 \sim \sigma^2 \chi^2_{N-M}$$

Parameter Variance / Standardized coefficient

standardized coefficient ("z-score"):

$$z_n := \frac{\hat{\beta}_n}{\widehat{\mathrm{se}}(\hat{\beta}_n)}, \quad \text{with } \widehat{\mathrm{se}}^2(\hat{\beta}_n) \text{ the } n\text{-th diagonal element of } (X^T X)^{-1}\hat{\sigma}^2$$

z_n would be $z_n \sim \mathcal{N}(0,1)$ if σ is known (under $H_0 : \beta_n = 0$).
With estimated $\hat{\sigma}$ it is $z_n \sim t_{N-M}$.

The Wald test for $H_0 : \beta_n = 0$ with size α is:

$$\text{reject } H_0 \text{ if } |z_n| = |\frac{\hat{\beta}_n}{\widehat{\mathrm{se}}(\hat{\beta}_n)}| > F^{-1}_{t_{N-M}}(1 - \frac{\alpha}{2})$$

i.e., its p-value is

$$p\text{-value}(H_0 : \beta_n = 0) = 2(1 - F_{t_{N-M}}(|z_n|)) = 2(1 - F_{t_{N-M}}(|\frac{\hat{\beta}_N}{\widehat{\mathrm{se}}(\hat{\beta}_N)}|))$$

and small p-values such as 0.01 and 0.05 are good.

Confidence interval

The $1-\alpha$ confidence interval for β_n:

$$\beta_n \pm F^{-1}_{t_{N-M}}(1-\frac{\alpha}{2})\ \widehat{\mathsf{se}}(\hat{\beta}_n)$$

For large N, $F_{t_{N-M}}$ converges to the standard normal cdf Φ.

As $\Phi^{-1}(1-\frac{0.05}{2}) \approx 1.95996 \approx 2$, the rule-of-thumb for a 5% confidence interval is

$$\beta_n \pm 2\,\widehat{\mathsf{se}}(\hat{\beta}_n)$$

Example

We have already fitted

$$\begin{aligned}\hat{y} =& \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \\ =& 5.583 + 0.779 x_1 - 1.699 x_2\end{aligned}$$

to the data:

x_1	x_2	y	$\hat{y}$	$\hat{\epsilon}^2 = (y - \hat{y})^2$
1	2	3	2.965	0.00122
2	3	2	2.045	0.00207
4	1	7	7.003	0.0000122
5	5	1	0.986	0.000196
RSS				0.00350

$$\hat{\sigma}^2 = \frac{1}{N-P}\sum_{n=1}^{N}\hat{\epsilon}_n^2 = \frac{1}{4-3}0.00350 = 0.00350$$

$$(X^T X)^{-1}\hat{\sigma}^2 = \begin{pmatrix} 0.00520 & -0.00075 & -0.00076 \\ -0.00075 & 0.00043 & -0.00020 \\ -0.00076 & -0.00020 & 0.00049 \end{pmatrix}$$

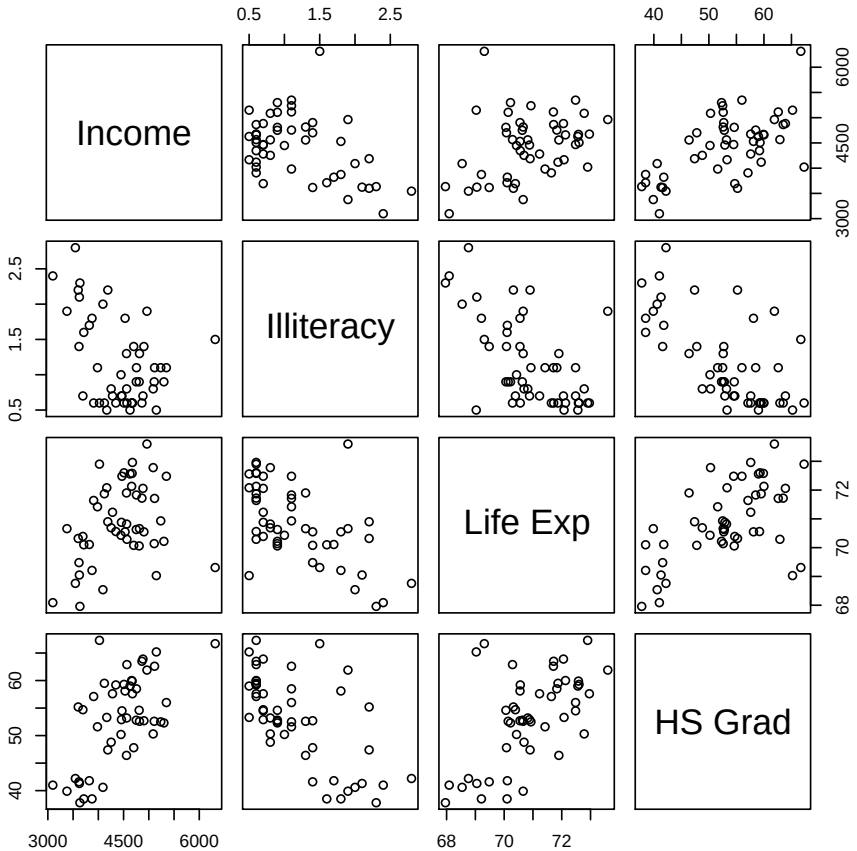
covariate	$\hat{\beta}_n$	$\widehat{\text{se}}(\hat{\beta}_n)$	z-score	p-value
(intercept)	5.583	0.0721	77.5	0.0082
X_1	0.779	0.0207	37.7	0.0169
X_2	−1.699	0.0221	−76.8	0.0083

Example 2

Example: sociographic data of the 50 US states in 1977.

state dataset:

- income (per capita, 1974),
- illiteracy (percent of population, 1970),
- life expectancy (in years, 1969–71),
- percent high-school graduates (1970).
- population (July 1, 1975)
- murder rate per 100,000 population (1976)
- mean number of days with minimum temperature below freezing (1931–1960) in capital or large city
- land area in square miles



Example 2

$$\begin{aligned}\text{Murder} =& \beta_0 + \beta_1 \text{Population} + \beta_2 \text{Income} + \beta_3 \text{Illiteracy} \\ &+ \beta_4 \text{LifeExp} + \beta_5 \text{HSGrad} + \beta_6 \text{Frost} + \beta_7 \text{Area}\end{aligned}$$

$N = 50$ states, $M = 8$ parameters, $N - M = 42$ degrees of freedom.

Least squares estimators:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	1.222e+02	1.789e+01	6.831	2.54e-08	***
Population	1.880e-04	6.474e-05	2.905	0.00584	**
Income	-1.592e-04	5.725e-04	-0.278	0.78232	
Illiteracy	1.373e+00	8.322e-01	1.650	0.10641	
‘Life Exp‘	-1.655e+00	2.562e-01	-6.459	8.68e-08	***
‘HS Grad‘	3.234e-02	5.725e-02	0.565	0.57519	
Frost	-1.288e-02	7.392e-03	-1.743	0.08867	.
Area	5.967e-06	3.801e-06	1.570	0.12391	

Outline

The Variable Selection Problem

Given a data set $\mathcal{D}^{\text{train}} \subseteq \mathbb{R}^M \times \mathcal{Y}$,
an error measure err,
a model class with a learning algorithm $\mathcal{A}$,

find the subset $V \subseteq \{1, 2, \ldots, M\}$ of (relevant) variables s.t. the model

$$\hat{y} := \mathcal{A}(\pi_V(\mathcal{D}^{\text{train}}))$$

learned on this subset V is best, i.e., for new test data $\mathcal{D}^{\text{test}}$ its test error

$$\text{err}(\hat{y}, \mathcal{D}^{\text{test}}),$$

is minimal.

Projection onto predictors V:

$$\pi_V(x, y) := (x_{i_1}, x_{i_2}, \ldots, x_{i_{\tilde{M}}}, y), \quad \text{for } V := \{i_1, i_2, \ldots, i_{\tilde{M}}\}$$

Greedy Search

- All 2^M subsets are too many to test (for larger M).
- Use a simple greedy search.
- **forward search:**
  - start with no variables.
  - test adding one more variable not yet in the model.
  - add the one leading to lowest validation error.
- **backward search:**
  - start with all variables.
  - test removing one more variable still in the model.
  - remove the one leading to lowest validation error.
- Does not guarantee to find the best variables subset.
  (But usually finds a useful one.)

Forward Search

procedure SELECTVARS-FORWARD($\mathcal{D}^{\text{train}\prime} \subseteq \mathbb{R}^M \times \mathcal{Y}, \text{err}, \mathcal{A}$)
  $(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}) := \text{split}(\mathcal{D}^{\text{train}\prime})$
  $V := \emptyset,\ v_{\text{best}} = \emptyset$
  $e_{\text{best}} := \text{err}(\mathcal{A}(\pi_V(\mathcal{D}^{\text{train}})), \pi_V(\mathcal{D}^{\text{val}}))$
  loop
    for $v \in \{1, 2, \ldots, M\} \setminus V$ **do**
      $V' := V \cup \{v\}$
      $\hat{y} := \mathcal{A}(\pi_{V'}(\mathcal{D}^{\text{train}}))$
      $e := \text{err}(\hat{y}, \pi_{V'}(\mathcal{D}^{\text{val}}))$
      if $e < e_{\text{best}}$ **then**
        $v_{\text{best}} := \{v\}$
        $e_{\text{best}} := e$
    if $v_{\text{best}} \subseteq V$ **then break**
    $V := V \cup v_{\text{best}}$
  return V

Backward Search

procedure SELECTVARS-BACKWARD($\mathcal{D}^{\text{train}\prime} \subseteq \mathbb{R}^M \times \mathcal{Y}, \text{err}, \mathcal{A}$)
  $(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}) := \text{split}(\mathcal{D}^{\text{train}\prime})$
  $V := \{1, 2, \ldots, M\}$, $v_{\text{best}} = \emptyset$
  $e_{\text{best}} := \text{err}(\mathcal{A}(\pi_V(\mathcal{D}^{\text{train}})), \pi_V(\mathcal{D}^{\text{val}}))$
  loop
    for $v \in V$ **do**
      $V' := V \setminus \{v\}$
      $\hat{y} := \mathcal{A}(\pi_{V'}(\mathcal{D}^{\text{train}}))$
      $e := \text{err}(\hat{y}, \pi_{V'}(\mathcal{D}^{\text{val}}))$
      if $e < e_{\text{best}}$ **then**
        $v_{\text{best}} := \{v\}$
        $e_{\text{best}} := e$
    if $V \cap v_{\text{best}} = \emptyset$ **then break**
    $V := V \setminus v_{\text{best}}$
  return V

Sequential Search with Variable Importance Heuristics

- Forward and backward search has to learn many models.
  - forward search: 1, 2, 3, ...
  - backward search: M, M-1, M-2, ...
- Further simplification: use a sequential search.
- Use a heuristics to assess **variable importance** once (without context)
  - e.g., the error of the single-variable model:

$$\text{imp}(m) := \text{err}(\mathcal{A}(\pi_{\{m\}}(\mathcal{D}^{\text{train}})), \mathcal{D}^{\text{val}})$$

- Add variables in order of increasing heuristics.
- Usually a full sequential sweep through all variables is done.
  - No difference between Forward and Backward Search.
- Faster, but even less reliable than forward/backward search.

Sequential Search

procedure SELECTVARS-SEQ($\mathcal{D}^{\text{train}\prime} \subseteq \mathbb{R}^M \times \mathcal{Y}$, err, $\mathcal{A}$, imp)
  $(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}) := \text{split}(\mathcal{D}^{\text{train}\prime})$
  $\mathcal{V} := \text{sort-increasing}(\{1, 2, \ldots, M\}, \text{imp})$
  $V := \emptyset$
  $e_{\text{best}} := \text{err}(\mathcal{A}(\pi_V(\mathcal{D}^{\text{train}})), \pi_V(\mathcal{D}^{\text{val}}))$
  for $m = 1, \ldots, M$ **do**
    $V := V \cup \{\mathcal{V}_m\}$
    $\hat{y} := \mathcal{A}(\pi_V(\mathcal{D}^{\text{train}}))$
    $e := \text{err}(\hat{y}, \pi_V(\mathcal{D}^{\text{val}}))$
    if $e < e_{\text{best}}$ **then**
      $e_{\text{best}} := e$
    else
      $V := V \setminus \{\mathcal{V}_m\}$
      break
  return V

Outline

Minimizing a Function via Coordinate Descent (CD)

Given a function $f : \mathbb{R}^N \to \mathbb{R}$, find x with minimal $f(x)$.

- Use the coordinate axes as descent direction
  - first x_1-axis, then x_2-axis, etc. (cyclic)
  - **one-dimensional subproblems**:

$$g_n(x) := \arg\min_{x_n \in \mathbb{R}} f(x_n; x_{-n}) := \arg\min_{x' \in \mathbb{R}} f(x_1, x_2, \ldots, x_{n-1}, x', x_{n+1}, \ldots, x_N)$$

- Coordinate Descent can be fast if solving the one-dimensional subproblems can be done analytically.
  - For smooth f, one needs to solve

$$\frac{\partial f(x_n; x_{-n})}{\partial x_n} \stackrel{!}{=} 0$$

  - Then also no step length is required !

Note: $x_{-n} := (x_1, \ldots, x_2, \ldots, x_{n-1}, x_{n+1}, \ldots, x_N)$ is the vector without element n for a vector $x \in \mathbb{R}^N$.

Coordinate Descent

1: **procedure**
MINIMIZE-CD($f : \mathbb{R}^N \to \mathbb{R}, g, x^{(0)} \in \mathbb{R}^N, i_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+$)
2: **for** $i := 1, \ldots, i_{\max}$ **do**
3: $x^{(i)} := x^{(i-1)}$
4: **for** $n := 1, \ldots, N$ **do**
5: $x_n^{(i)} := g_n(x_{-n}^{(i)})$
6: **if** $f(x^{(i-1)}) - f(x^{(i)}) < \epsilon$ **then**
7: **return** $x^{(i)}$
8: **error** "not converged in $i_{\max}$ iterations"

with

g : solvers g_n for the n-th one-dimensional subproblem

$$g_n(x_1, x_2, \ldots, x_{n-1}, x_{n+1}, \ldots, x_N) := \arg\min_{x' \in \mathbb{R}} f(x_1, \ldots, x_{n-1}, x', x_{n+1}, \ldots, x_N)$$

Example: Simple Quadratic Function

Minimize

$$f(x_1, x_2) := x_1^2 + x_2^2 + x_1 x_2$$

One dimensional problem for x_1:

$$\begin{aligned} f(x_1; x_2) =& x_1^2 + x_2^2 + x_1 x_2 \\ \frac{\partial f}{\partial x_1}(x_1; x_2) =& 2x_1 + x_2 \stackrel{!}{=} 0 \\ \rightsquigarrow\ & x_1 = -\frac{1}{2} x_2 \\ \text{i.e., } & g_1(x_2) := -\frac{1}{2} x_2 \end{aligned}$$

and analogous for x_2:

$$g_2(x_1) := -\frac{1}{2} x_1$$

Example: Simple Quadratic Function

Minimize

$$f(x_1, x_2) := x_1^2 + x_2^2 + x_1 x_2, \quad x^{(0)} := (1, 1)$$

$$g_1(x_2) := -\frac{1}{2} x_2, \quad g_2(x_1) := -\frac{1}{2} x_1$$

i	$x^{(i)}$ before	n	$g_n(x^{(i)})$	$x^{(i)}$ after
1	$(1, 1)$	1	$-1/2$	$(-1/2, 1)$
	$(-1/2, 1)$	2	$1/4$	$(-1/2, 1/4)$
2	$(-1/2, 1/4)$	1	$-1/8$	$(-1/8, 1/4)$
	$(-1/8, 1/4)$	2	$1/16$	$(-1/8, 1/16)$
$\vdots$				

Note: Minimize $f(x_1, x_2) := x_1^2 + x_2^2$ via CD yourself. What is different? Why?

Learn Linear Regression via CD

Minimize

$$f(\hat{\beta}) := ||y - X\hat{\beta}||^2 \propto \hat{\beta}^T X^T X \hat{\beta} - 2y^T X \hat{\beta}$$

$$\begin{aligned} f(\hat{\beta}_m; \hat{\beta}_{-m}) &= x_m^T x_m \hat{\beta}_m^2 + 2\hat{\beta}_{-m}^T X_{-m}^T x_m \hat{\beta}_m + \hat{\beta}_{-m}^T X_{-m}^T X_{-m} \hat{\beta}_{-m} \\ &\quad - 2y^T x_m \hat{\beta}_m - 2y^T X_{-m} \hat{\beta}_{-m} \\ &\propto x_m^T x_m \hat{\beta}_m^2 - 2(y - X_{-m}\hat{\beta}_{-m})^T x_m \hat{\beta}_m \end{aligned}$$

$$\frac{\partial f(\hat{\beta}_m; \hat{\beta}_{-m})}{\partial \hat{\beta}_m} \stackrel{!}{=} 0 \rightsquigarrow \hat{\beta}_m = \frac{(y - X_{-m}\hat{\beta}_{-m})^T x_m}{x_m^T x_m}$$

Note: $x_m := X_{.,m}$ denotes the m-th column of X,
X_{-m} denotes the matrix X without column m.

Learn Linear Regression via CD

1: **procedure** LEARN-LINREG-CD($\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}, i_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+$)

2: $X := (x_1, x_2, \ldots, x_N)^T$

3: $y := (y_1, y_2, \ldots, y_N)^T$

4: $\hat{\beta}_0 := (0, \ldots, 0)$

5: $\hat{\beta} :=$ MINIMIZE-CD($f(\hat{\beta}) := (y - X\hat{\beta})^T(y - X\hat{\beta})$, $g(\hat{\beta}_m; \hat{\beta}_{-m}) := \frac{(y - X_{-m}\hat{\beta}_{-m})^T x_m}{x_m^T x_m}$, $\hat{\beta}_0, \alpha, i_{\max}, \epsilon$)

6: **return** $\hat{\beta}$

Note: $x_m := X_{\cdot,m}$ denotes the m-th column of X,
X_{-m} denotes the matrix X without column m.

Outline

L1 Regularization

Let X the predictor matrix and y the target vector,
$\hat{\theta}$ the model parameters,
$\hat{y}$ the model predictions and
ℓ the loss/error.

L2 regularization:

$$f(\hat{\theta}) := \ell(y, \hat{y}(\hat{\theta}, X)) + \lambda ||\hat{\theta}||_2^2 = \ldots + \lambda \sum_{p=1}^{P} \hat{\theta}_p^2$$

L1 regularization:

$$f(\hat{\theta}) := \ell(y, \hat{y}(\hat{\theta}, X)) + \lambda ||\hat{\theta}||_1 = \ldots + \lambda \sum_{p=1}^{P} |\hat{\theta}_p|$$

Why L1 Regularization?

$$\begin{aligned} \text{min.} \quad & f(\hat{\theta}) := \ell(y, \hat{y}(\hat{\theta}, X)) + \lambda ||\hat{\theta}||_1 \\ & \hat{\theta} \in \mathbb{R}^P \end{aligned}$$

is equivalent to

$$\begin{aligned} \text{min.} \quad & f(\hat{\theta}) := \ell(y, \hat{y}(\hat{\theta}, X)) \\ & ||\hat{\theta}||_1 \leq B \\ & \hat{\theta} \in \mathbb{R}^P \end{aligned}$$

with

$$B := ||\hat{\theta}^*||_1$$

Note: $\hat{\theta}^*$ denotes the optimal parameters. Thus this equivalence provides insight, but cannot (yet) be used to solve the problem.

Why L1 Regularization?

$$\min_{\hat{\theta} \in \mathbb{R}^P} f(\hat{\theta}) := \ell(y, \hat{y}(\hat{\theta}, X)) + \lambda ||\hat{\theta}||_1$$

is equivalent to

$$\min_{\hat{\theta} \in \mathbb{R}^P} f(\hat{\theta}) := \ell(y, \hat{y}(\hat{\theta}, X))$$
$$||\hat{\theta}||_1 \leq B$$

with

$$B := ||\hat{\theta}^*||_1$$

$$\min_{\hat{\theta} \in \mathbb{R}^P} f(\hat{\theta}) := \ell(y, \hat{y}(\hat{\theta}, X)) + \lambda ||\hat{\theta}||_2^2$$

is equivalent to

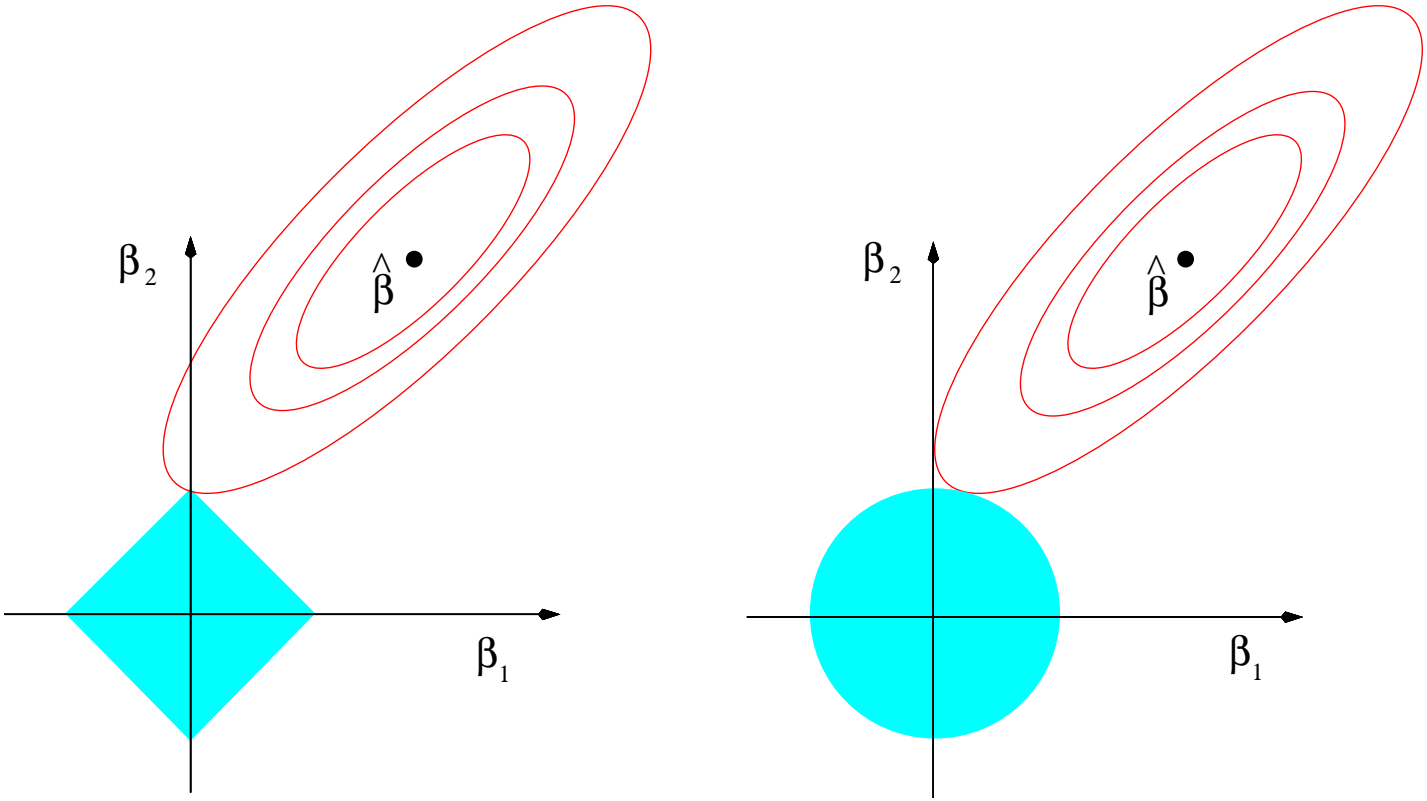
$$\min_{\hat{\theta} \in \mathbb{R}^P} f(\hat{\theta}) := \ell(y, \hat{y}(\hat{\theta}, X))$$
$$||\hat{\theta}||_2^2 \leq B$$

with

$$B := ||\hat{\theta}^*||_2^2$$

Note: $\hat{\theta}^*$ denotes the optimal parameters. Thus this equivalence provides insight, but cannot (yet) be used to solve the problem.

Why L1 Regularization?



source: [?, p. 90]

Regularized Linear Regression

Let X the predictor matrix and y the target vector,
$\hat{\beta}$ the linear regression model parameters,
$\hat{y} := X\hat{\beta}$ the linear regression model predictions and
$\ell(y, \hat{y}) := ||y - \hat{y}||_2^2$ the RSS loss/error.

L2 Regularized Linear Regression (**Ridge Regression**):

$$\begin{aligned} f(\hat{\beta}) &:= \ell(y, \hat{y}(\hat{\beta}, X)) + \lambda||\hat{\beta}||_2^2 \\ &\propto \hat{\beta}^T X^T X \hat{\beta} - 2y^T X \hat{\beta} + \lambda \hat{\beta}^T \hat{\beta} \\ &= \hat{\beta}^T (X^T X + \lambda I)\hat{\beta} - 2y^T X \hat{\beta} \end{aligned}$$

- L2 regularized problem has same structure as unregularized one.
- All learning algorithms work seamlessly.

Regularized Linear Regression

Let X the predictor matrix and y the target vector,
$\hat{\beta}$ the linear regression model parameters,
$\hat{y} := X\hat{\beta}$ the linear regression model predictions and
$\ell(y, \hat{y}) := ||y - \hat{y}||_2^2$ the RSS loss/error.

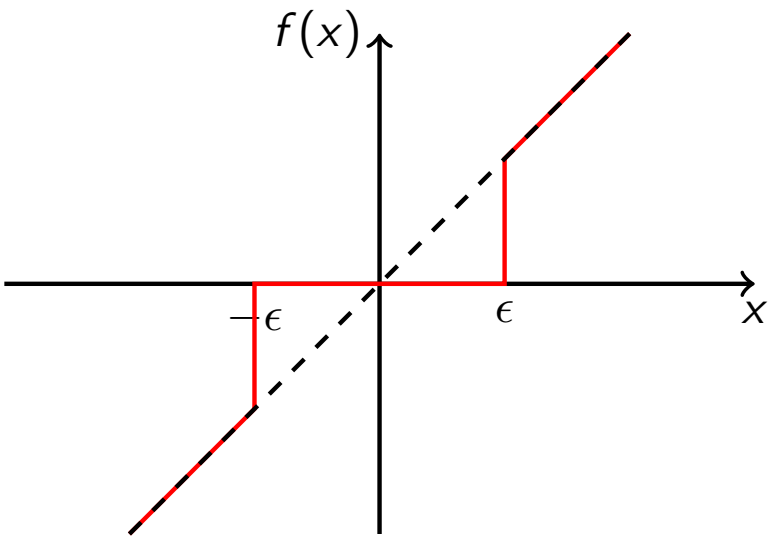
L1 regularized Linear Regression (**Lasso**):

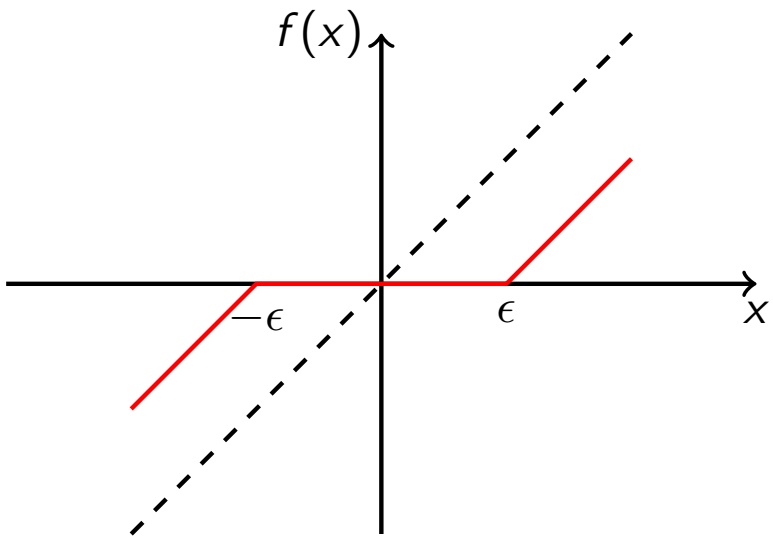
$$f(\beta) := \ell(y, \hat{y}) + \lambda||\beta||_1$$

$$\propto \hat{\beta}^T X^T X\hat{\beta} - 2y^T X\hat{\beta} + \lambda \sum_{m=1}^{M} |\beta_m|$$

- L1 regularized problem has new terms $|\beta_m|$.
  - Esp. non-differentiable at 0.
- All learning algorithms seen so far do not work.
  - Solving SLE is not applicable.
  - Gradient Descent does not work.

Hard & Soft Thresholding





$$\text{hard}(x,\epsilon) := \begin{cases} x, & \text{if } |x| > \epsilon \\ 0, & \text{else} \end{cases}$$

$$\text{soft}(x,\epsilon) := \begin{cases} x-\epsilon, & \text{if } x > \epsilon \\ 0, & \text{if } |x| \leq \epsilon \\ x+\epsilon, & \text{if } x < -\epsilon \end{cases}$$

Coordinate Descent for L1 Regularized Linear Regression

$$f(\hat{\beta}) := \hat{\beta}^T X^T X \hat{\beta} - 2y^T X \hat{\beta} + \lambda \sum_{m=1}^{M} |\beta_m|$$

$$f(\hat{\beta}_m; \hat{\beta}_{-m}) \propto x_m^T x_m \hat{\beta}_m^2 - 2(y - X_{-m}\hat{\beta}_{-m})^T x_m \hat{\beta}_m + \lambda|\beta_m|$$

$$\frac{\partial f(\hat{\beta}_m; \hat{\beta}_{-m})}{\partial \hat{\beta}_m} \stackrel{!}{=} 0 \rightsquigarrow \hat{\beta}_m = \begin{cases} \frac{(y - X_{-m}\hat{\beta}_{-m})^T x_m - \frac{1}{2}\lambda}{x_m^T x_m}, & \hat{\beta}_m > 0 \\ \frac{(y - X_{-m}\hat{\beta}_{-m})^T x_m + \frac{1}{2}\lambda}{x_m^T x_m}, & \hat{\beta}_m < 0 \end{cases}$$

$$\rightsquigarrow \hat{\beta}_m = \text{soft}\left(\frac{(y - X_{-m}\hat{\beta}_{-m})^T x_m}{x_m^T x_m}, \frac{\frac{1}{2}\lambda}{x_m^T x_m}\right)$$

Note: LASSO = Least Absolute Selection and Shrinkage Operator.

Learn L1-regularized Linear Regression via CD (Shooting Algorithm)

1: **procedure** LEARN-LINREG-L1REG-CD($\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}, \lambda \in \mathbb{R}^+, i_{\max} \in \mathbb{N}, \epsilon \in \mathbb{R}^+$)

2: $X := (x_1, x_2, \ldots, x_N)^T$

3: $y := (y_1, y_2, \ldots, y_N)^T$

4: $\hat{\beta}_0 := (0, \ldots, 0)$

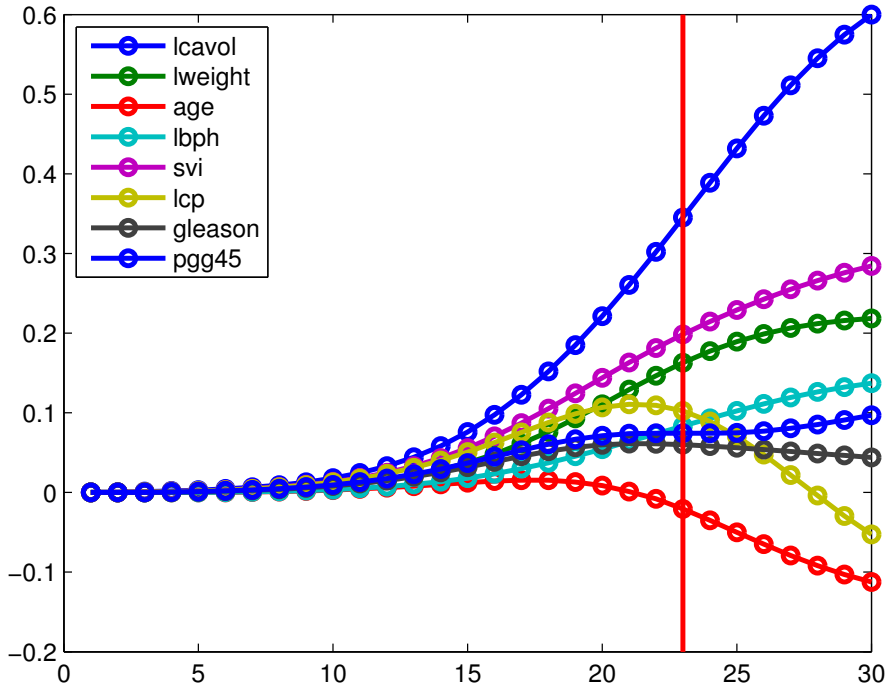
5: $\hat{\beta} :=$ MINIMIZE-CD($f(\hat{\beta}) := (y - X\hat{\beta})^T(y - X\hat{\beta}) + \lambda||\beta||_1$, $g(\hat{\beta}_m; \hat{\beta}_{-m}) := \text{soft}(\frac{(y - X_{-m}\hat{\beta}_{-m})^T x_m}{x_m^T x_m}, \frac{\frac{1}{2}\lambda}{x_m^T x_m})$, $\hat{\beta}_0, \alpha, i_{\max}, \epsilon$)

6: **return** $\hat{\beta}$

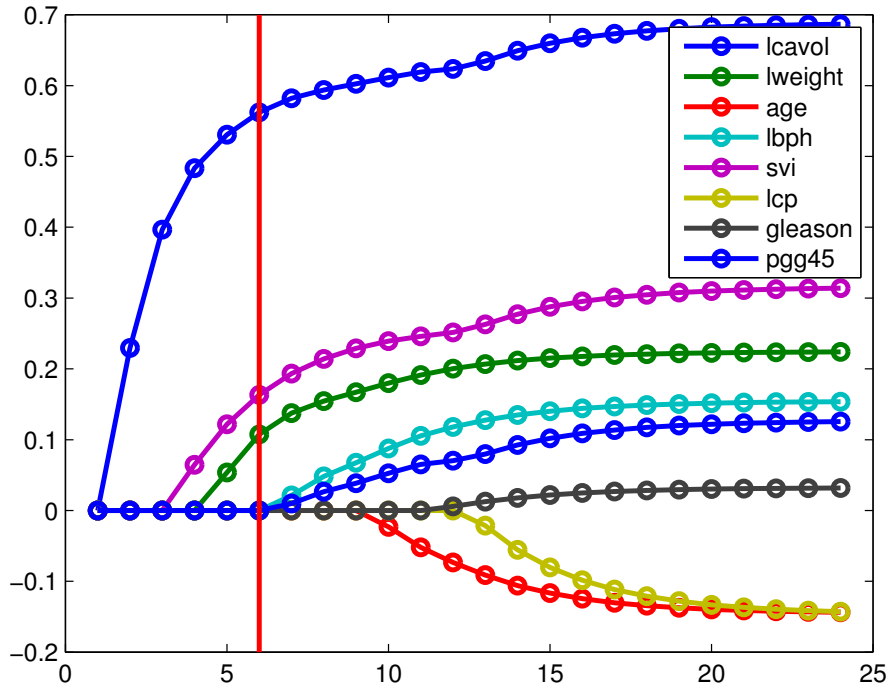
Note: $x_m := X_{\cdot,m}$ denotes the m-th column of X, X_{-m} denotes the matrix X without column m.

Regularization Paths

L2 regularization



L1 regularization



x-axis: bound B on parameter size.
y-axis: parameter $\hat{\theta}$.

source: [**?**, p. 437]

Summary

- **High-dimensional data** poses problems as many parameters have to be estimated from comparable few instances.
- Non-linear effects can be captured by **derived predictor variables**.
  - e.g., in **polynomial models**.
  - making even originally low-dimensional data high-dimensional.
- Relevant variables can be searched explicitly through a greedy **forward search** and **backward search**.
- To minimize a function, **coordinate descent** cyclicly chooses the coordinate axes as descent direction.
  - efficient, if the **one-dimensional subproblems** can be solved analytically.
  - does need no step length.

Further Readings

- [**?**, chapter 6], [**?**, chapter 13], [**?**, chapter 3.3–8].

References