Machine Learning

A. Supervised Learning
A.4. High-Dimensional Data

Lars Schmidt-Thieme, Nicolas Schilling

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

Outline

1. Variable Interactions and Polynomial Models
2. Variable Selection via Forward and Backward Search
3. Minimizing a Function via Coordinate Descent
4. 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

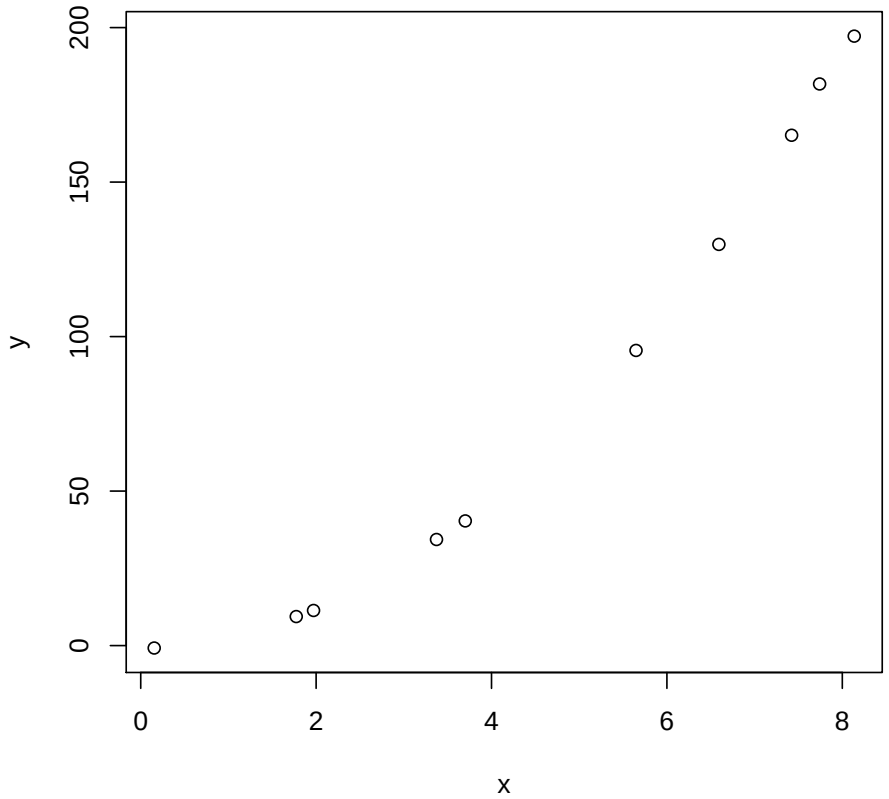
Assume a target variable does not depend linearly on a predictor variable, but say quadratic.

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.

Consider the way length s of a moving object with velocity v and duration t: the way length s of a moving object vs. its constant velocity v and duration t:

$$s = vt + \epsilon$$

- additional $1s$ duration will change the way length not in a uniform way
- high impact for large velocities
- little impact for small velocities

v and t are said to interact: s 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):

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

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):

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

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

Polynomial Models

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

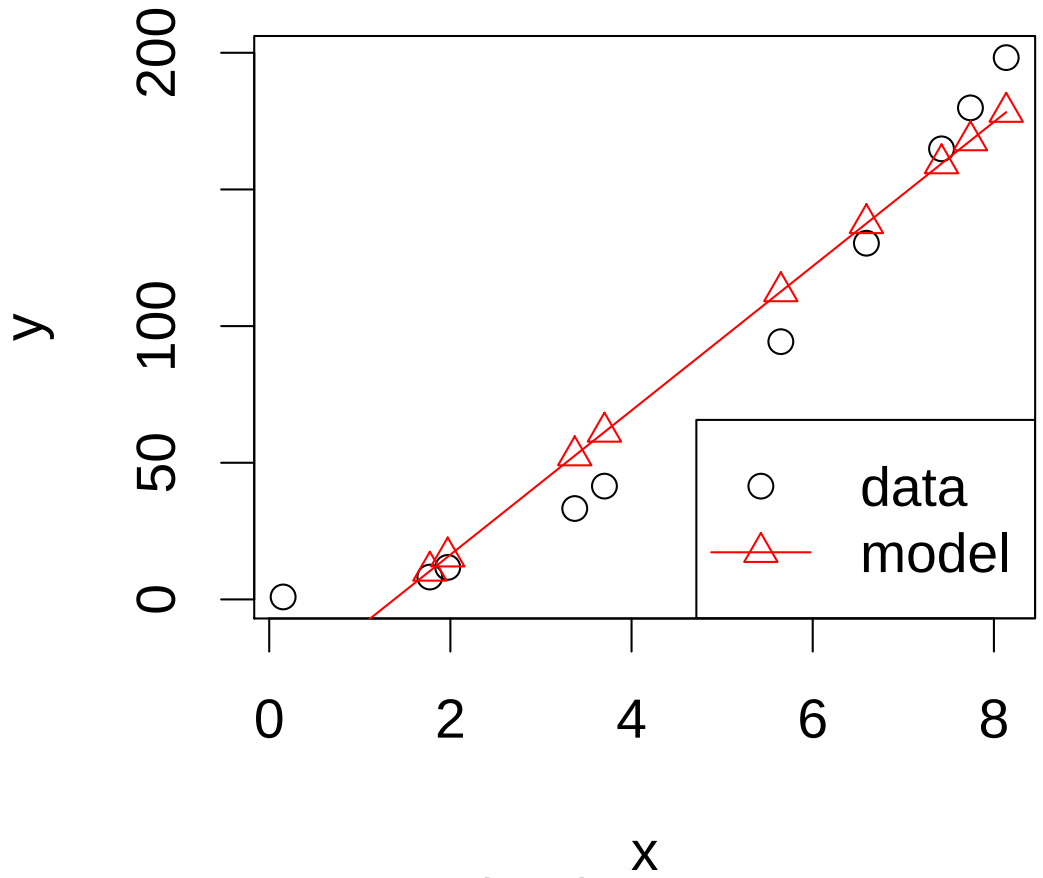
$$\hat{y}(x) := \hat{\beta}_0 + \sum_{m=1}^{M} \hat{\beta}_m x_m \qquad \text{degree 1}$$

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

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

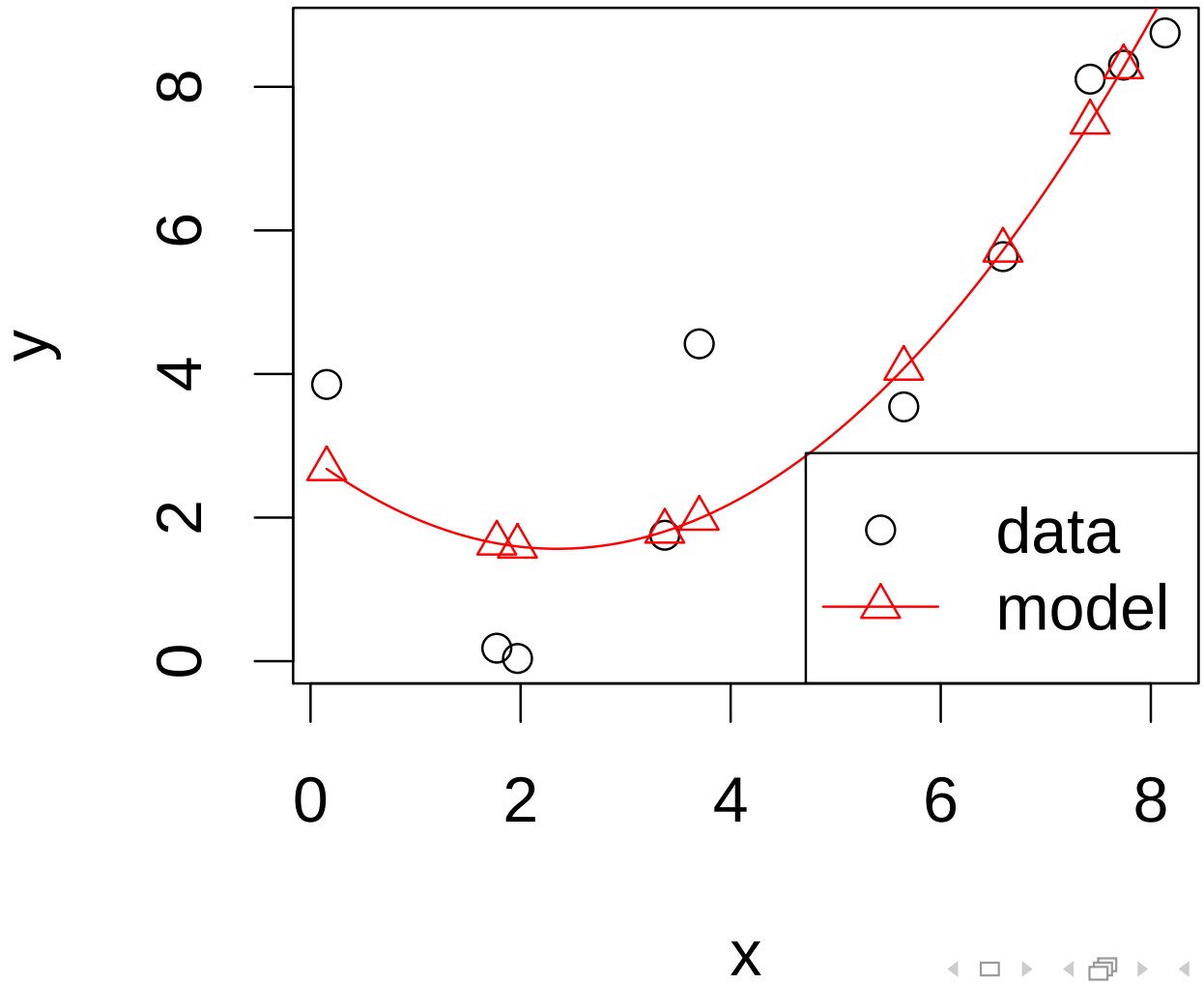
$$+ \sum_{m_1=1}^{M} \sum_{m_2=m_1}^{M} \cdots \sum_{m_d=m_{d-1}}^{M} \hat{\theta}_{m_1,m_2,\ldots,m_d} x_{m_1} x_{m_2} \cdots x_{m_d} \qquad \text{degree } d$$

High Polynomial Degress, High Model Complexity

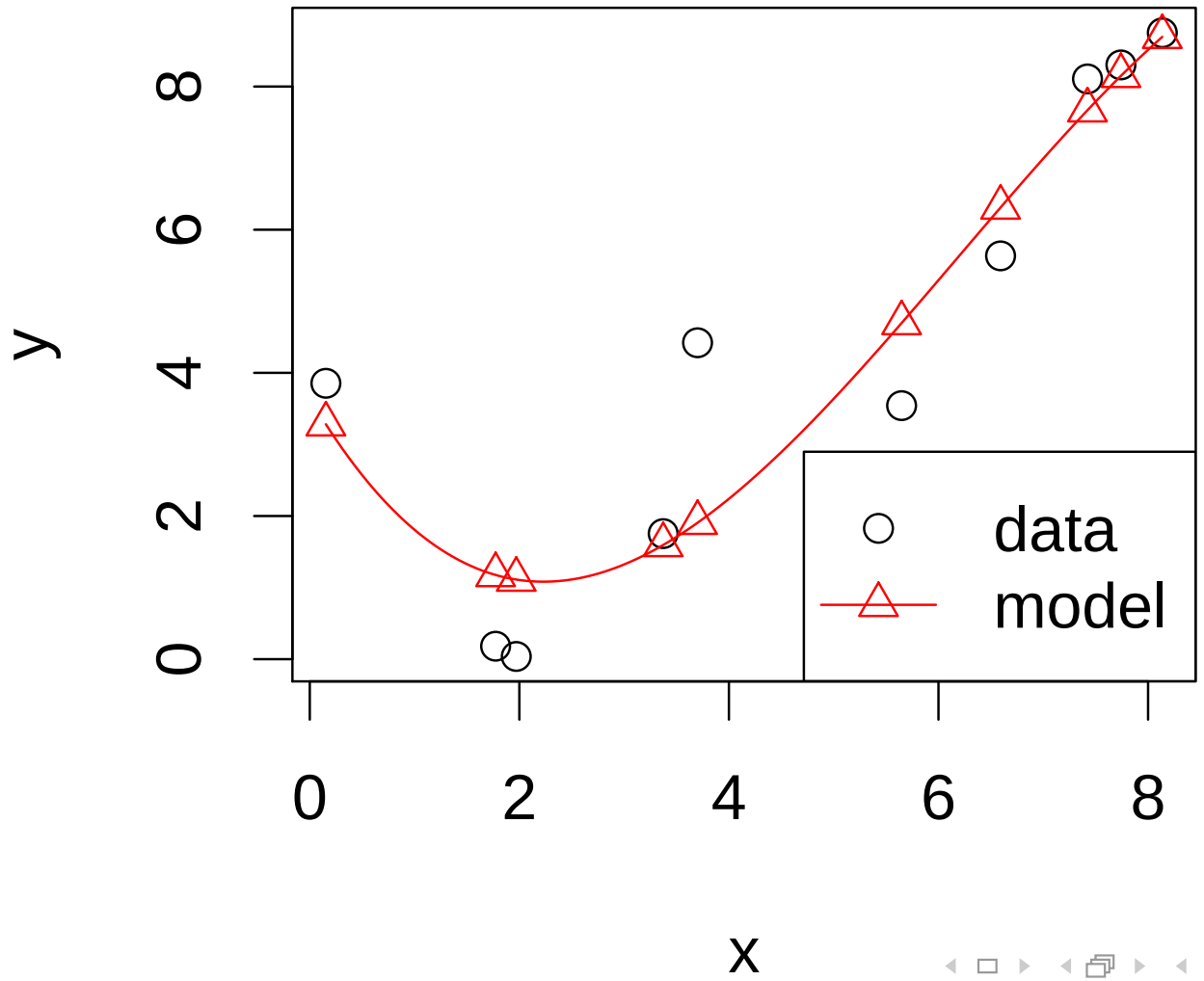


If a model does not well explain the data,
e.g., if the true model is quadratic, but we try to fit a linear model,
one says, the model **underfits**.

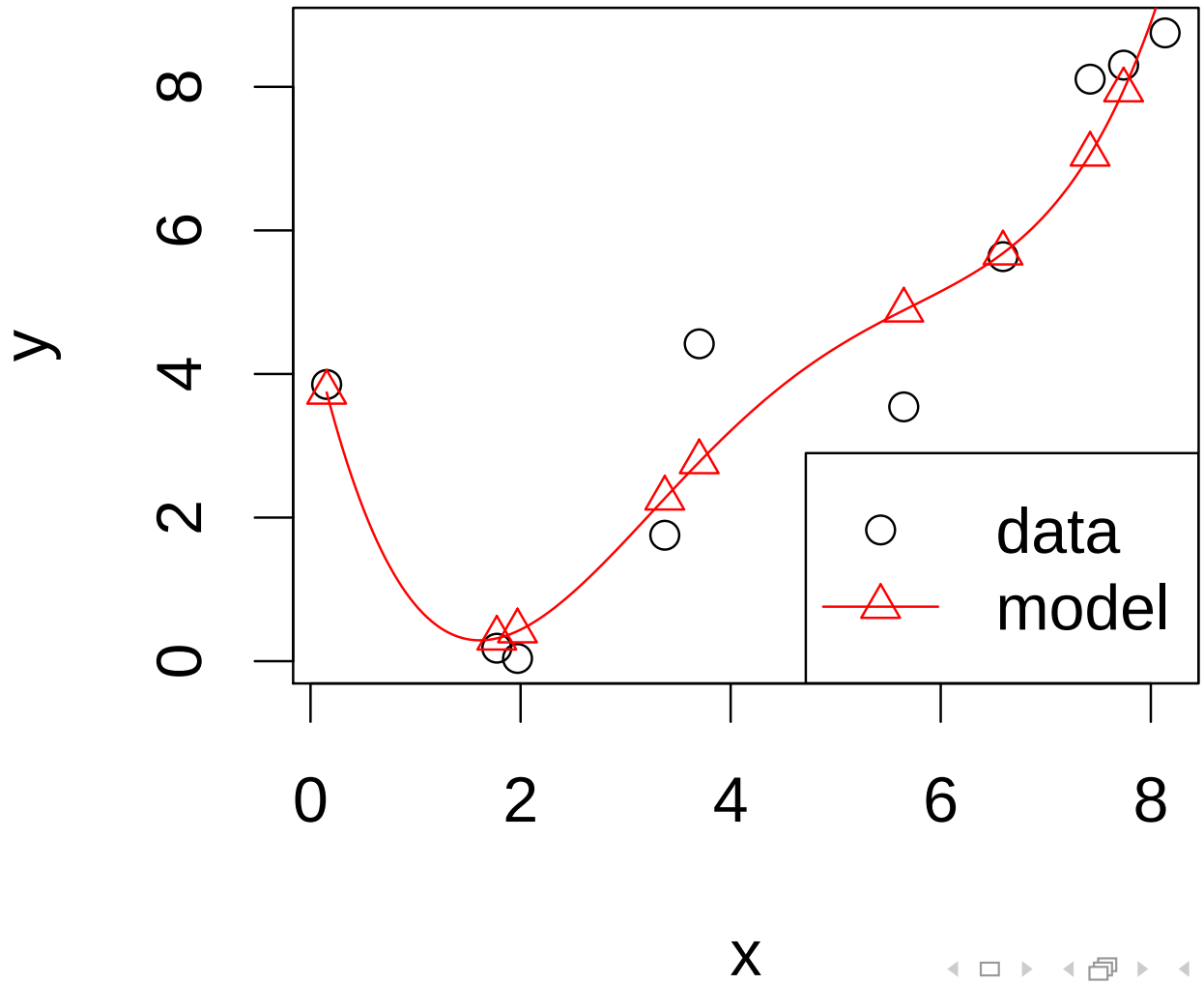
High Polynomial Degress, High Model Complexity



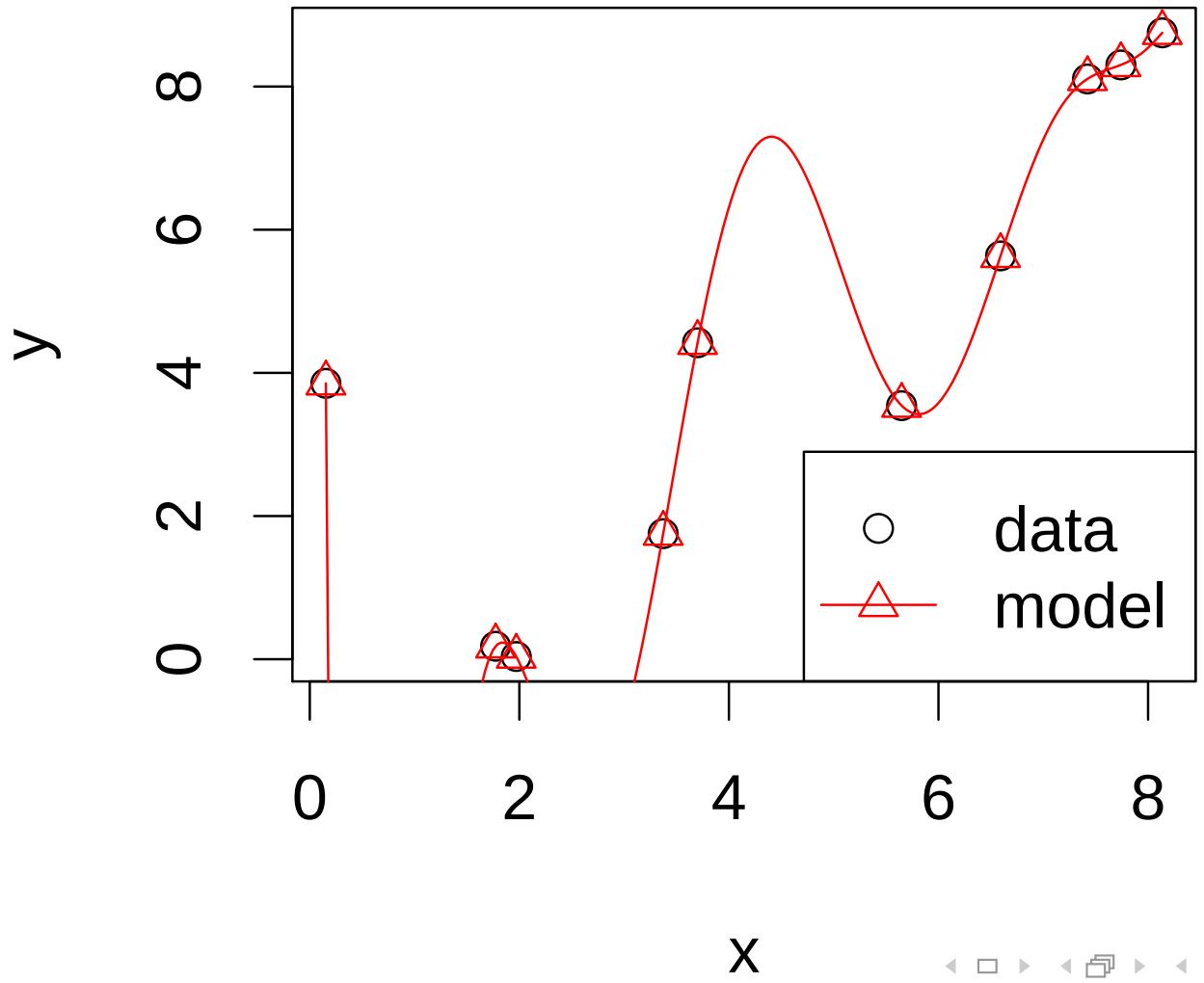
High Polynomial Degress, High Model Complexity



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

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_{n-1} x^{n-1} \tag{1}$$

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 value of x.)

As the polynomial

$$\hat{y}(x) = \sum_{i=1}^{n} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

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

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 $\mathcal{D}^{\text{test}}$ it's 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($\mathcal{D}^{\text{train}\prime} \subseteq \mathbb{R}^M \times \mathcal{Y}, \text{err}, \mathcal{A}$)
  $(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}) := \text{split}(\mathcal{D}^{\text{train}\prime})$
  $V := \emptyset$
  $e_{\text{allbest}} := \text{err}(\mathcal{A}(\pi_V(\mathcal{D}^{\text{train}})), \pi_V(\mathcal{D}^{\text{val}}))$
  $v_{\text{best}} := 1$
  while $v_{\text{best}} \neq 0$ **do**
    $v_{\text{best}} := 0$
    $e_{\text{best}} := e_{\text{allbest}}$
    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 $e_{\text{best}} < e_{\text{allbest}}$ **then**
      $V := V \cup \{v_{\text{best}}\}$
      $e_{\text{allbest}} := e_{\text{best}}$
  return V

Backward Search

1: **procedure** SELECTVARS-BACKWARD($\mathcal{D}^{\text{train}\prime} \subseteq \mathbb{R}^M \times \mathcal{Y}, \text{err}, \mathcal{A}$)
2: $(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}) := \text{split}(\mathcal{D}^{\text{train}\prime})$
3: $V := \{1, 2, \ldots, M\}$
4: $e_{\text{allbest}} := \text{err}(\mathcal{A}(\pi_V(\mathcal{D}^{\text{train}})), \pi_V(\mathcal{D}^{\text{val}}))$
5: $v_{\text{best}} := 1$
6: **while** $v_{\text{best}} \neq 0$ **do**
7: $v_{\text{best}} := 0$
8: $e_{\text{best}} := e_{\text{allbest}}$
9: **for** $v \in V$ **do**
10: $V' := V \backslash \{v\}$
11: $\hat{y} := \mathcal{A}(\pi_{V'}(\mathcal{D}^{\text{train}}))$
12: $e := \text{err}(\hat{y}, \pi_{V'}(\mathcal{D}^{\text{val}}))$
13: **if** $e < e_{\text{best}}$ **then**
14: $v_{\text{best}} := v$
15: $e_{\text{best}} := e$
16: **if** $e_{\text{best}} < e_{\text{allbest}}$ **then**
17: $V := V \backslash \{v_{\text{best}}\}$
18: $e_{\text{allbest}} := e_{\text{best}}$
19: **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 simplication: 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)
  $(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{val}}) := \text{split}(\mathcal{D}^{\text{train}\prime})$
  $\mathcal{V} := \text{sort-increasing}(\{1, 2, \ldots, M\}, \text{imp})$
  $V := \emptyset$
  $e_{\text{best}} := \text{err}(\mathcal{A}(\pi_V(\mathcal{D}^{\text{train}})), \pi_V(\mathcal{D}^{\text{val}}))$
  $m_{\text{best}} := 1$
  for $m = 1, \ldots, M$ **do**
    $v := \mathcal{V}_m$
    $V := V \cup \{v\}$
    $\hat{y} := \mathcal{A}(\pi_V(\mathcal{D}^{\text{train}}))$
    $e := \text{err}(\hat{y}, \pi_V(\mathcal{D}^{\text{val}}))$
    if $e < e_{\text{best}}$ **then**
      $m_{\text{best}} := m$
      $e_{\text{best}} := e$
  $V := \{1, 2, \ldots, m_{\text{best}}\}$
  return V

Outline

Minimizing a Function via Coordinate Descent (CD)

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

- Use the coordinate axes as descent direction
  - first β_1-axis, then β_2-axis, etc. (cyclic)
  - **one-dimensional subproblems**:

$$g_n(\beta) := \arg\min_{\beta_n \in \mathbb{R}} f(\beta_n; \beta_{-n}) := \arg\min_{\beta' \in \mathbb{R}} f(\beta_1, \ldots, \beta_{n-1}, \beta', \beta_{n+1}, \ldots, \beta_N)$$

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

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

  - Then also no step length is required !

Note: $\beta_{-n} := (\beta_1, \ldots, \beta_2, \ldots, \beta_{n-1}, \beta_{n+1}, \ldots, \beta_N)$ is the vector without the n-th element for a vector $\beta \in \mathbb{R}^N$.

Coordinate Descent

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

g solves g_n for the n-th one-dimensional subproblem

$$g_n(\beta_1, \ldots, \beta_{n-1}, \beta_{n+1}, \ldots, \beta_N) := \arg\min_{\beta' \in \mathbb{R}} f(\beta_1, \ldots, \beta_{n-1}, \beta', \beta_{n+1}, \ldots, \beta_N)$$

Example: Simple Quadratic Function

Minimize

$$f(\beta_1, \beta_2) := \beta_1^2 + \beta_2^2 + \beta_1\beta_2$$

One dimensional problem for β_1:

$$\begin{aligned} f(\beta_1; \beta_2) =& \beta_1^2 + \beta_2^2 + \beta_1\beta_2 \\ \frac{\partial f}{\partial \beta_1}(\beta_1; \beta_2) =& 2\beta_1 + \beta_2 \stackrel{!}{=} 0 \end{aligned}$$

$$\rightsquigarrow \beta_1 = -\frac{1}{2}\beta_2$$

$$\text{i.e., } g_1(\beta_2) := -\frac{1}{2}\beta_2$$

and analogous for β_2:

$$g_2(\beta_1) := -\frac{1}{2}\beta_1$$

Example: Simple Quadratic Function

Minimize

$$f(\beta_1, \beta_2) := \beta_1^2 + \beta_2^2 + \beta_1\beta_2, \quad \beta^{(0)} := (1, 1)$$

$$g_1(\beta_2) := -\frac{1}{2}\beta_2, \quad g_2(\beta_1) := -\frac{1}{2}\beta_1$$

i	$\beta^{(i)}$ before	n	$g_n(\beta^{(i)})$	$\beta^{(i-1)}$ 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 $f(\beta_1, \beta_2) := \beta_1^2 + \beta_2^2$ 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: $x_m := X_{\cdot,m}$ 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($f(\hat{\beta}) := (y - X\hat{\beta})^T(y - X\hat{\beta})$, $g(\hat{\beta}_m; \hat{\beta}_{-m}) := \frac{(y - X_{-m}\hat{\beta}_{-m})^T x_m}{x_m^T x_m}$, $\hat{\beta}_0, i_{\max}, \epsilon$)

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

L2 regularization:

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

L1 regularization:

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

Why L1 Regularization?

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

is equivalent to

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

with

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

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

Why L1 Regularization?

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

is equivalent to

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

with

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

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

is equivalent to

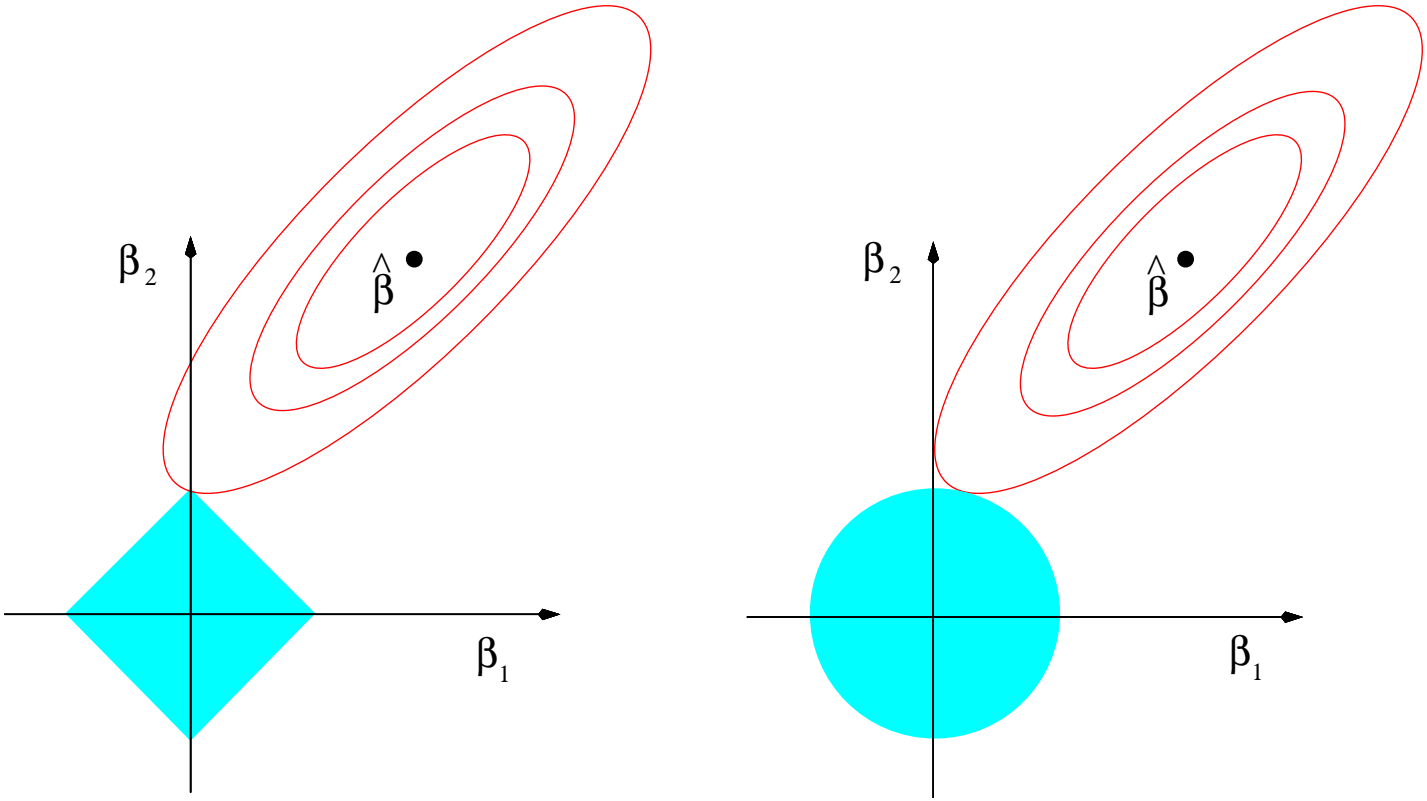
$$\min_{\hat{\beta} \in \mathbb{R}^P} f(\hat{\beta}) := \ell(y, \hat{y}(\hat{\beta}, X))$$
$$||\hat{\beta}||_2^2 \leq B$$

with

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

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

Why L1 Regularization?



source: [HTFF05, 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**):

$$\begin{aligned} f(\hat{\beta}) &:= \ell(y, \hat{y}(\hat{\beta}, X)) + \lambda||\hat{\beta}||_2^2 \\ &\propto \hat{\beta}^T X^T X \hat{\beta} - 2y^T X \hat{\beta} + \lambda \hat{\beta}^T \hat{\beta} \\ &= \hat{\beta}^T (X^T + \lambda^{\frac{1}{2}} I)(X + \lambda^{\frac{1}{2}} I)\hat{\beta} - 2y^T X \hat{\beta} \end{aligned}$$

- 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.

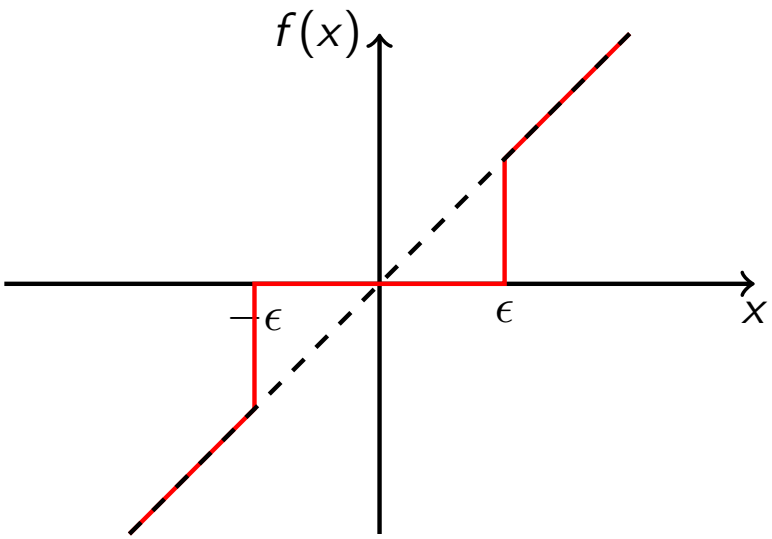
L1 regularized Linear Regression (**Lasso**):

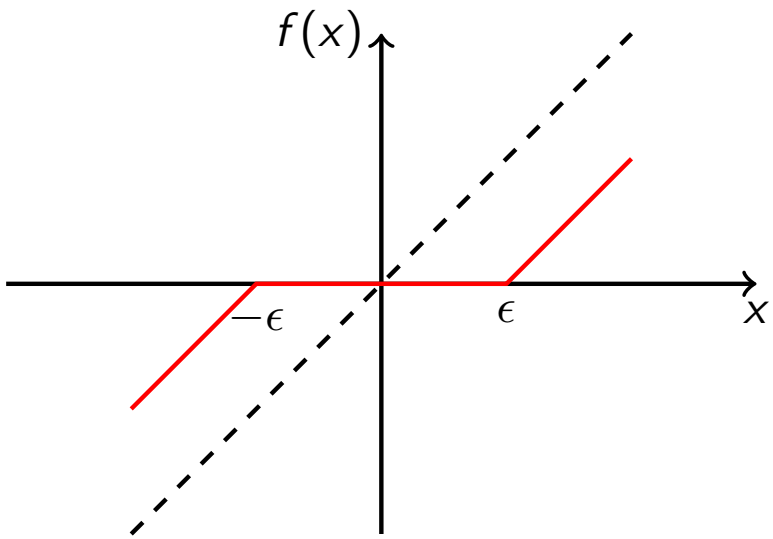
$$f(\beta) := \ell(y, \hat{y}) + \lambda||\beta||_1$$

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

- 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 Gradient for L1 Regularized Linear Regression

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

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

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

$$\hat{\beta}_m = \frac{(y - X_{-m}\hat{\beta}_{-m})^T x_m + \frac{1}{2}\lambda}{x_m^T x_m}, \quad \hat{\beta}_m < 0$$

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

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

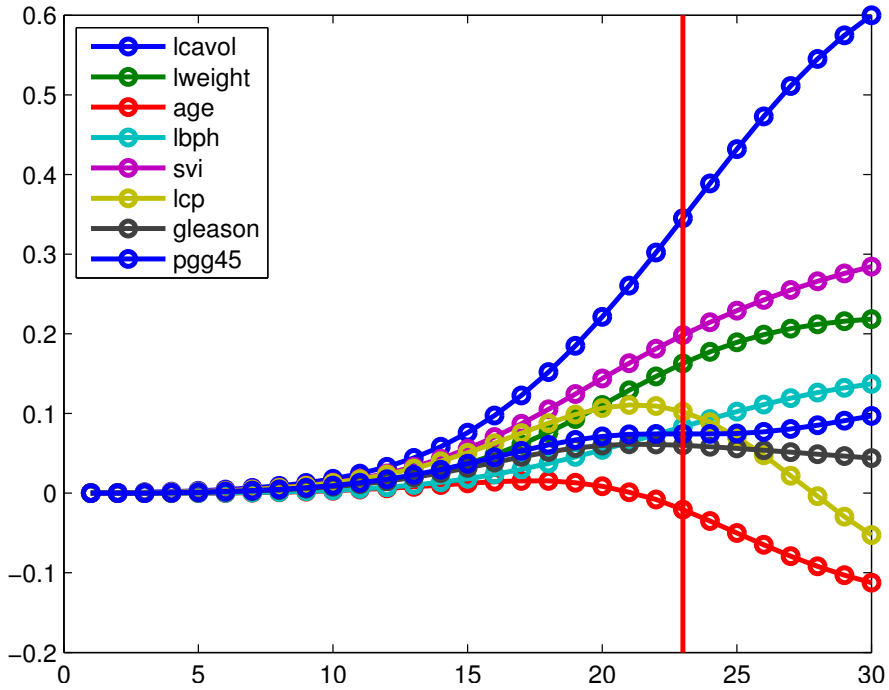
5: $\hat{\beta} :=$ MINIMIZE-CD($f(\hat{\beta}) := (y - X\hat{\beta})^T(y - X\hat{\beta}) + \lambda||\beta||_1$, $g(\hat{\beta}_m; \hat{\beta}_{-m}) := \text{soft}(\frac{(y - X_{-m}\hat{\beta}_{-m})^T x_m}{x_m^T x_m}, \frac{\frac{1}{2}\lambda}{x_m^T x_m})$, $\hat{\beta}_0, \alpha, i_{\max}, \epsilon$)

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

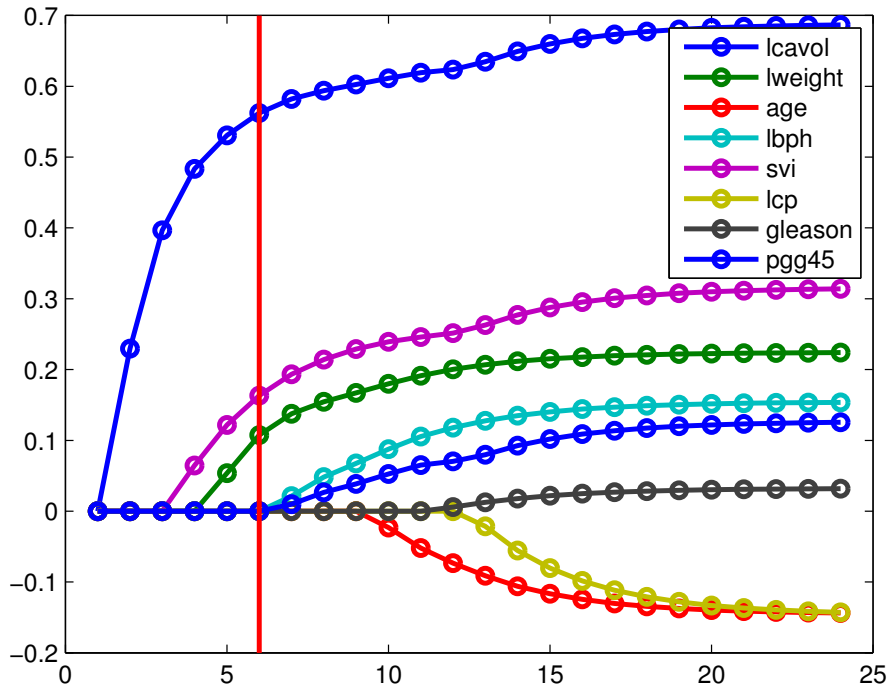
Note: $x_m := X_{\cdot,m}$ denotes the m-th column of X, X_{-m} denotes the matrix X without column m.

Regularization Paths

L2 regularization



L1 regularization



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

source: [Mur12, 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.
- Variable selection also can be accomplished by **L1 regularization**.
  - **L1 regularized linear regression (LASSO)** can be learned by coordinate descent (**shooting algorithm**).

Further Readings

- [JWHT13, chapter 6], [Mur12, chapter 13], [HTFF05, chapter 3.3–8].

References

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin.
The elements of statistical learning: data mining, inference and prediction, volume 27.
2005.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani.
An introduction to statistical learning.
Springer, 2013.

Kevin P. Murphy.
Machine learning: a probabilistic perspective.
The MIT Press, 2012.