Chapter 7

Fitting in other Norms

In this chapter we discuss methods for parameter estimation in cases, where the definition of "best" is different from the (weighted) least squares, cf Section 5.1. More precisely, we are given a vector function $\boldsymbol{r} : \mathbb{R}^n \mapsto \mathbb{R}^m$ and we want to find $\hat{\boldsymbol{x}}$ that minimizes some measure of $\boldsymbol{r}(\boldsymbol{x})$, for instance $\|\boldsymbol{r}(\boldsymbol{x})\|_1$ or $\|\boldsymbol{r}(\boldsymbol{x})\|_\infty$. These two cases are treated in the first two sections, and in the rest of the chapter we discuss a method which can be thought of as a hybrid between the least squares and the least absolute deviation definitions of "best".

The function $\boldsymbol{r}$ may depend nonlinearly on $\boldsymbol{x}$, and we present algorithms of Gauss–Newton type, cf Section 6.1, based on successive approximations by first order Taylor expansions,

$$\begin{aligned} & r_i(\boldsymbol{x}+\boldsymbol{h}) \;\simeq\; \ell_i(\boldsymbol{h}) \;\equiv\; r_i(\boldsymbol{x}) + \boldsymbol{\nabla} r_i(\boldsymbol{x})^T \boldsymbol{h} \;, \quad i = 1, \ldots, m \;, \\ \Updownarrow & \\ & \boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h}) \;\simeq\; \boldsymbol{\ell}(\boldsymbol{h}) \;\equiv\; \boldsymbol{r}(\boldsymbol{x}) + \boldsymbol{J}(\boldsymbol{x})\boldsymbol{h} \;, \end{aligned} \tag{7.1}$$

where $\boldsymbol{\nabla} r_i \in \mathbb{R}^n$ is the gradient of r_i and $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ is the Jacobian of $\boldsymbol{r}$. It has the rows $\boldsymbol{J}_{i:} = (\boldsymbol{\nabla} r_i)^T$. In the first two sections we combine the Gauss–Newton method with a trust region approach, cf Section 2.4. The generic algorithm is

Algorithm 7.2. Generic algorithm

Given starting point $\boldsymbol{x}$ and trust region radius Δ
while not STOP
  $\hat{\boldsymbol{h}} = \operatorname{argmin}_{\|\boldsymbol{h}\|_\infty \le \Delta} L(\boldsymbol{h})$
  $\varrho = \big((f(\boldsymbol{x}) - f(\boldsymbol{x}+\hat{\boldsymbol{h}})\big) / \big(L(\boldsymbol{0}) - L(\hat{\boldsymbol{h}})\big)$
  if $\varrho \ge 0.75$ **then** $\Delta := \Delta * 2$
  if $\varrho \le 0.25$ **then** $\Delta := \Delta / 3$
  if $\varrho > 0$ **then** $\boldsymbol{x} := \boldsymbol{x} + \hat{\boldsymbol{h}}$
end

The functions f and L are defined by $f(\boldsymbol{x}) = \|\boldsymbol{r}(\boldsymbol{x})\|_p$ and $L(\boldsymbol{h}) = \|\boldsymbol{\ell}(\boldsymbol{h})\|_p$, respectively. Combining these definitions with (7.1) we see that $L(\boldsymbol{0}) = f(\boldsymbol{x})$.

7.1. Fitting in the L_∞ norm

We seek a minimizer $\hat{\boldsymbol{x}}$ of the function

$$f(\boldsymbol{x}) \;=\; \|\boldsymbol{r}(\boldsymbol{x})\|_\infty \;=\; \max_i |r_i(\boldsymbol{x})| \;.$$

Hald and Madsen [25] proposed to use Algorithm 7.2. The linearized problem

$$\hat{\boldsymbol{h}} \;=\; \underset{\|\boldsymbol{h}\|_\infty \le \Delta}{\operatorname{argmin}} \; \left\{ \, L(\boldsymbol{h}) \equiv \|\boldsymbol{r}(\boldsymbol{x}) + \boldsymbol{J}(\boldsymbol{x})\,\boldsymbol{h}\|_\infty \, \right\}$$

is solved by an LP algorithm, cf Appendix A.9. To get an LP formulation of the problem, we introduce the variable $h_{n+1} = L(\boldsymbol{h})$ and the extended vector

$$\tilde{\boldsymbol{h}} \;=\; \begin{pmatrix} \boldsymbol{h} \\ h_{n+1} \end{pmatrix} \in \mathbb{R}^{n+1} \;.$$

Now $\hat{\boldsymbol{h}}$ can be found by solving the problem

$$\begin{aligned} &\text{minimize} \quad h_{n+1} \\ &\text{subject to} \quad \left.\begin{aligned} -\Delta \le h_i &\le \Delta \\ -h_{n+1} \le r_i(\boldsymbol{x}) + \boldsymbol{\nabla} r_i(\boldsymbol{x})^T \boldsymbol{h} &\le h_{n+1} \end{aligned}\right\} \quad i = 1, \ldots, n \end{aligned}$$

See [25] for further information. This simple linear programming formulation is enabled by the change of the trust region definition from the 2–norm discussed in Section 2.4 to the ∞–norm used here.

7.2. Fitting in the $\mathbf{L}_1$ norm

We seek a minimizer $\hat{\boldsymbol{x}}$ of the function

$$f(\boldsymbol{x}) \;=\; \|\boldsymbol{r}(\boldsymbol{x})\|_1 \;=\; \sum_{i=1}^{m} |r_i(\boldsymbol{x})| \;.$$

In some applications the solution is referred to as the *least absolute deviation* fit.

Hald and Madsen [26] proposed to use Algorithm 7.2. The linearized problem

$$\hat{\boldsymbol{h}} \;=\; \underset{\|\boldsymbol{h}\|_\infty \le \Delta}{\operatorname{argmin}} \; \left\{ \, L(\boldsymbol{h}) \equiv \|\boldsymbol{r}(\boldsymbol{x}) + \boldsymbol{J}(\boldsymbol{x})\,\boldsymbol{h}\|_1 \, \right\}$$

is solved by an LP algorithm: We introduce auxiliary variables h_{n+i}, $i = 1, \ldots, m$, and the extended vector

$$\tilde{\boldsymbol{h}} = \begin{pmatrix} \boldsymbol{h} \\ h_{n+1} \\ \vdots \\ h_{n+m} \end{pmatrix} \in \mathbb{R}^{n+m} \; .$$

Now $\hat{\boldsymbol{h}}$ can be found by solving the problem

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{m} h_{n+i} \\ &\text{subject to} \quad \left.\begin{aligned} -\Delta \le h_i &\le \Delta \\ -h_{n+i} \le r_i(\boldsymbol{x}) + \boldsymbol{\nabla} r_i(\boldsymbol{x})^T \boldsymbol{h} &\le h_{n+i} \end{aligned}\right\} \quad i = 1, \ldots, m \end{aligned}$$

See [26] for further information.

Example 7.1. When $\boldsymbol{r}$ is an affine function, $\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{y} - \boldsymbol{F}\boldsymbol{x}$, the gradients are constant, and the LP formulation is

$$\begin{aligned} &\text{minimize} \quad \textstyle\sum_{i=1}^{m} x_{n+i} \\ &\text{subject to} \quad -x_{n+i} \le y_i - \boldsymbol{F}_{i,:}\,\boldsymbol{x} \le x_{n+i} \; , \quad i = 1, \ldots, m \; . \end{aligned}$$

This formulation is the background for the *Barrodale–Roberts* algorithm [2], which is often used for robust parameter estimation with linear fitting models, cf Examples 5.5 and 7.3. ∎

7.3. Huber estimation

This approach combines the smoothness of the least squares estimator with the robustness of the L_1-estimator, cf Example 5.5.

For a function $\boldsymbol{r} : \mathbb{R}^n \mapsto \mathbb{R}^m$ the *Huber estimator* $\boldsymbol{x}_\gamma$ is defined by[1)]

$$\boldsymbol{x}_\gamma \;=\; \underset{\boldsymbol{x}}{\operatorname{argmin}}\big\{ f_\gamma(\boldsymbol{x}) \;\equiv\; \sum_{i=1}^{m} \phi_\gamma(r_i(\boldsymbol{x}))\big\} \;, \tag{7.3}$$

where ϕ_γ is the *Huber function*,

$$\phi_\gamma(u) \;=\; \begin{cases} \dfrac{1}{2\gamma} u^2 & \text{if } |u| \le \gamma \;, \\ |u| - \frac{1}{2}\,\gamma & \text{if } |u| > \gamma \;. \end{cases} \tag{7.4}$$

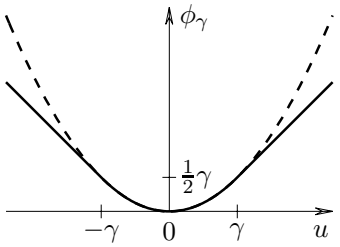

Figure 7.1. *Huber function (full line) and the scaled* L_2 *function* $\frac{1}{2\gamma}u^2$ *(dotted line).*

The threshold γ is used to distinguish between "small" and "large" function values (residuals). Based on the values of the $r_i(\boldsymbol{x})$ we define a generalized sign vector $\boldsymbol{s} = \boldsymbol{s}_\gamma(\boldsymbol{x})$ and an "activity matrix" $\boldsymbol{W} = \boldsymbol{W}_\gamma(\boldsymbol{x}) = \operatorname{diag}(w_1, \ldots, w_m)$. Note that $w_i = 1 - s_i^2$.

	$r_i(\boldsymbol{x}) < -\gamma$	$\lvert r_i(\boldsymbol{x})\rvert \le \gamma$	$r_i(\boldsymbol{x}) > \gamma$
$s_i(\boldsymbol{x})$	-1	0	1
$w_i(\boldsymbol{x})$	0	1	0

(7.5)

Now the objective function in (7.3) can be expressed as

$$f_\gamma(\boldsymbol{x}) \;=\; \frac{1}{2\gamma}\, \boldsymbol{r}^T \boldsymbol{W}\, \boldsymbol{r} + \boldsymbol{r}^T \boldsymbol{s} - \tfrac{1}{2}\,\gamma\, \boldsymbol{s}^T \boldsymbol{s} \;, \tag{7.6}$$

where we have omitted the argument $(\boldsymbol{x})$ and index γ on the right-hand side. The gradient is

$$\boldsymbol{\nabla} f_\gamma \;=\; \frac{1}{\gamma}\, \boldsymbol{J}^T (\boldsymbol{W}\, \boldsymbol{r} + \boldsymbol{\gamma}\, \boldsymbol{s}) \;. \tag{7.7}$$

[1)] Strictly speaking, it is misleading to discuss this approach under the heading "other norms": f_γ in (7.3) is not a norm: the triangle inequality is not satisfied.

At a minimizer the gradient must be zero. Thus, a necessary condition for $\boldsymbol{x}_\gamma$ being a minimizer of f_γ is that it satisfies the equation

$$\boldsymbol{J}(\boldsymbol{x})^T \left(\boldsymbol{W}_\gamma(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x}) + \gamma\, \boldsymbol{s}_\gamma(\boldsymbol{x})\right) \;=\; \boldsymbol{0}\;. \tag{7.8}$$

7.3.1. Linear Huber estimation

First consider an affine function

$$\boldsymbol{r}(\boldsymbol{x}) \;=\; \boldsymbol{y} - \boldsymbol{F}\,\boldsymbol{x}\;, \tag{7.9}$$

where $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{F} \in \mathbb{R}^{m\times n}$ are constant. It follows from (7.5) that $s_i(\boldsymbol{x}) = 0$ for all $\boldsymbol{x}$ between the two hyperplanes $r_i(\boldsymbol{x}) = -\gamma$ and $r_i(\boldsymbol{x}) = \gamma$. These $2m$ hyperplanes divide R^n into subregions $\{D_k\}$. Inside each subregion $\boldsymbol{s}_\gamma(\boldsymbol{x})$ and $\boldsymbol{W}_\gamma(\boldsymbol{x})$ are constant, so f_γ is a *piecewise quadratic*. The gradient is

$$\boldsymbol{\nabla} f_\gamma(\boldsymbol{x}) \;=\; -\frac{1}{\gamma}\,\boldsymbol{F}^T\big(\boldsymbol{W}(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{F}\,\boldsymbol{x}) + \gamma\,\boldsymbol{s}(\boldsymbol{x})\big)\;. \tag{7.10}$$

It varies continuously across the hyperplanes, while the Hessian

$$\boldsymbol{\nabla}^2 f_\gamma(\boldsymbol{x}) \;=\; \frac{1}{\gamma}\,\boldsymbol{F}^T\boldsymbol{W}(\boldsymbol{x})\,\boldsymbol{F} \;\equiv\; \boldsymbol{H}(\boldsymbol{x}) \tag{7.11}$$

is constant in the interior of each D_k and jumps as $\boldsymbol{x}$ crosses one of the dividing hyperplanes.

Example 7.2. Consider the affine function $\boldsymbol{r} : \mathbb{R}^2 \mapsto \mathbb{R}^3$ given by

$$\boldsymbol{r}(\boldsymbol{x}) \;=\; \begin{pmatrix} 1.5 \\ 2.0 \\ -3.5 \end{pmatrix} - \begin{pmatrix} -0.5 & 2.0 \\ 3.0 & -1.0 \\ -1.0 & 0.5 \end{pmatrix} \boldsymbol{x}\;,$$

and let the threshold $\gamma = 0.5$.

In $\mathbb{R}^2$ a hyperplane is a straight line, and Figure 7.2 below shows the lines, along which $r_i(\boldsymbol{x}) = 0$ (full lines); the dividing hyperplanes, $r_i(\boldsymbol{x}) = \pm\gamma$ (dotted lines) and two of the subregions are indicated by hatching. For $\boldsymbol{x} \in D_k$ we have $\boldsymbol{s} = \begin{pmatrix} 0 & -1 & -1 \end{pmatrix}^T$, while $\boldsymbol{s} = \begin{pmatrix} -1 & 1 & -1 \end{pmatrix}^T$ for $\boldsymbol{x} \in D_j$.

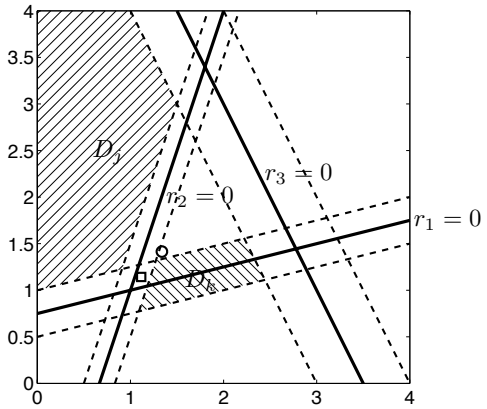

Figure 7.2. *Dividing hyperplanes in* $\mathbb{R}^2$.

The square indicates the Huber solution for $\gamma = 0.5$, $\boldsymbol{x}_\gamma = \begin{pmatrix} 1.116 & 1.143 \end{pmatrix}^T$ with $\boldsymbol{s}_\gamma = \begin{pmatrix} 0 & 0 & -1 \end{pmatrix}^T$. For comparison, the circle marks the least squares solution to $\boldsymbol{r}(\boldsymbol{x}) \simeq \boldsymbol{0}$: $\boldsymbol{x}_{(2)} = \begin{pmatrix} 1.337 & 1.415 \end{pmatrix}^T$. ∎

With a slight modification of (5.13) we can show that the Hessian (7.11) is positive semidefinite. This implies that if we find a stationary point $\boldsymbol{x}_\mathrm{s}$, ie $\boldsymbol{\nabla} f_\gamma(\boldsymbol{x}_\mathrm{s}) = \boldsymbol{0}$, then it is a minimizer for f_γ: $\boldsymbol{x}_\gamma = \boldsymbol{x}_\mathrm{s}$. We use Newton's method with line search to find a stationary point.

Algorithm 7.12. Linear Huber
Given starting point $\boldsymbol{x}$ and threshold γ
repeat
  Find $\boldsymbol{v}$ by solving $\boldsymbol{H}(\boldsymbol{x})\boldsymbol{v} = -\boldsymbol{\nabla} f_\gamma(\boldsymbol{x})$
  $\boldsymbol{x} := \text{line\_search}(\boldsymbol{x}, \boldsymbol{v})$
until STOP

The line search is very simple: $\varphi(\alpha) = f_\gamma(\boldsymbol{x} + \alpha\boldsymbol{v})$ is a piecewise quadratic in the scalar variable α, so α^* is a root for φ', which is piecewise linear; the coefficients change when $\boldsymbol{x} + \alpha\boldsymbol{v}$ passes a dividing hyperplane for increasing α. The stopping criterion is that $\boldsymbol{s}_\gamma(\boldsymbol{x}+\alpha^*\boldsymbol{v}) = \boldsymbol{s}_\gamma(\boldsymbol{x})$. See [33] and [38] about details and efficient implementation.

Example 7.3. The function (7.9) may be the residual from data fitting with a linear model, cf Chapter 5. *Peter Huber* [29] introduced his M-estimator as a method for reducing the effect of wild points, cf Example 5.5, and from (7.8) we see that this is achieved by replacing a large r_i by just the threshold γ times the *sign* of this wild point residual. ∎

Example 7.4. Suppose that γ is changed to δ and that $\boldsymbol{s}_\delta(\boldsymbol{x}_\delta) = \boldsymbol{s}_\gamma(\boldsymbol{x}_\gamma) = \boldsymbol{s}$, and therefore $\boldsymbol{W}_\delta(\boldsymbol{x}_\delta) = \boldsymbol{W}_\gamma(\boldsymbol{x}_\gamma) = \boldsymbol{W}$. Then it follows from (7.8) and (7.10) that the Huber solution is a linear function of the threshold,

$$\boldsymbol{x}_\delta \;=\; \boldsymbol{x}_\gamma + (\gamma - \delta)\left(\boldsymbol{F}^T\boldsymbol{W}\,\boldsymbol{F}\right)^{-1}\boldsymbol{F}^T\boldsymbol{s}\;. \tag{7.13}$$

Let $\boldsymbol{x}_{(2)}$ denote the least squares solution to $\boldsymbol{r}(\boldsymbol{x}) \simeq \boldsymbol{0}$. This is also the Huber solution (with $\boldsymbol{s} = \boldsymbol{0}$) for all $\gamma \geq \gamma_\infty \equiv \|\boldsymbol{r}(\boldsymbol{x}_{(2)})\|_\infty$. For $\gamma < \gamma_\infty$ one or more r_i will be "large", so $\boldsymbol{s}$ changes.

Madsen and Nielsen [34] showed that there exists a $\gamma_0 > 0$ such that $\boldsymbol{s}_\gamma(\boldsymbol{x}^*_\gamma)$ is constant for $\gamma \leq \gamma_0$ and used this together with (7.13) to develop an efficient method for linear L_1 estimation. This might replace the LP algorithm used to solve the linearized problems in Section 7.2. ∎

7.3.2. Nonlinear Huber Estimation

Now consider a function $\boldsymbol{r}$ that depends nonlinearly on $\boldsymbol{x}$. We must use iteration to find the minimizer $\boldsymbol{x}_\gamma$ of f_γ. At the current iterate $\boldsymbol{x}$ we use the approximation (7.1) and the corresponding approximation to the objective function

$$\begin{aligned} f_\gamma(\boldsymbol{x}+\boldsymbol{h}) \;&\simeq\; L(\boldsymbol{h}) \\ &=\; \frac{1}{2\gamma}\,\boldsymbol{\ell}^T\boldsymbol{W}\,\boldsymbol{\ell} + \boldsymbol{\ell}^T\boldsymbol{s} - \tfrac{1}{2}\,\gamma\,\boldsymbol{s}^T\boldsymbol{s}\;, \end{aligned}$$

where $\boldsymbol{\ell} = \boldsymbol{\ell}(\boldsymbol{h}) = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{J}(\boldsymbol{x})\boldsymbol{h}$ and $\boldsymbol{s}$ and $\boldsymbol{W}$ are given by (7.5) with $\boldsymbol{r}(\boldsymbol{x})$ replaced by $\boldsymbol{\ell}(\boldsymbol{h})$.

As in Sections 7.1 and 7.2 we can combine this Gauss–Newton model with a trust region approach, cf [27]. Instead we shall describe a Levenberg–Marquardt like algorithm, cf 6.18, see Algorithm 7.14.

The linearized problem at 1° is solved by a slight modification of Algorithm 7.12 with starting point $\boldsymbol{v} = \boldsymbol{0}$. Note that $L(\boldsymbol{0}) = f_\gamma(\boldsymbol{x})$.

Algorithm 7.14. Nonlinear Huber
Given starting point $\boldsymbol{x}$, threshold γ and $\mu > 0$. $\nu = 2$
while not STOP
  $\hat{\boldsymbol{h}} = \operatorname{argmin}_{\boldsymbol{h}} \left\{ L(\boldsymbol{h}) + \frac{1}{2}\mu h^T h \right\}$ {1°}
  $\varrho := (f_\gamma(\boldsymbol{x}) - f_\gamma(\boldsymbol{x}+\hat{\boldsymbol{h}}))/(L(\boldsymbol{0}) - L(\hat{\boldsymbol{h}}))$
  if $\varrho > 0$ {step acceptable}
    $\boldsymbol{x} := \boldsymbol{x} + \hat{\boldsymbol{h}}$
    $\mu := \mu * \max\{\frac{1}{3}, 1 - (2\varrho - 1)^3\}$; $\nu := 2$
  else
    $\mu := \mu * \nu$; $\nu := 2 * \nu$
  end
 end
end

Finally, we give two examples of modified Huber functions with reduced influence from positive components of $\boldsymbol{r}$.

$$\widehat{\phi}_\gamma(u) = \begin{cases} |u| - \frac{1}{2}\gamma & \text{for } u < -\gamma \ , \\ \dfrac{1}{2\gamma}\, u^2 & \text{for } -\gamma \le u \le 0 \ , \\ 0 & \text{for } u > 0 \ . \end{cases}$$

$$\widetilde{\phi}_\gamma(u) = \begin{cases} \dfrac{1}{2\gamma}\, u^2 & \text{for } u \le \gamma \ , \\ u - \frac{1}{2}\gamma & \text{for } u > \gamma \ . \end{cases}$$

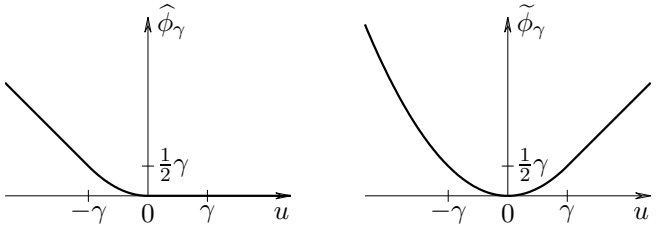

Figure 7.3. *One-sided Huber functions.one-sided Huber function*

See [39] about implementation and some applications of these functions.