CHAPTER

25 ROBUST REGRESSION

CHAPTER CONTENTS

Although LS is useful in various practical applications, it is sensitive to *outliers*, samples containing large noise. In this chapter, alternative learning methods that possess high robustness against outliers are introduced.

25.1 NONROBUSTNESS OF ℓ_2-LOSS MINIMIZATION

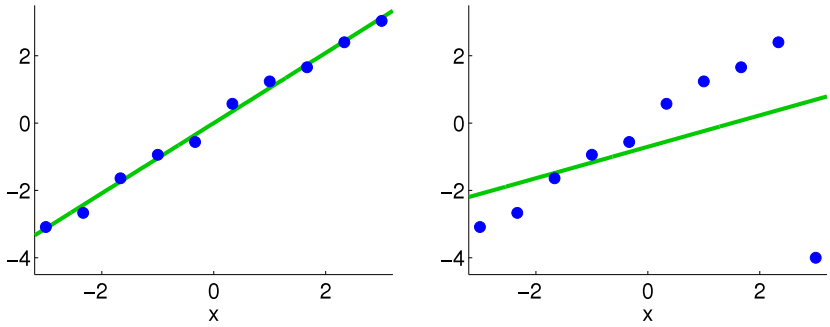
A LS solution for straight-line model $f_{\boldsymbol{\theta}}(x) = \theta_1 + \theta_2 x$ obtained from 10 training samples is illustrated in Fig. 25.1. If there is no outlier as in Fig. 25.1(a), a good solution can be obtained by LS. However, if there exists an outlier as in Fig. 25.1(b), the solution is strongly affected by the outlier.

When a large number of training samples are handled, it would be natural to consider that more or less outliers are included. In such a situation, LS is less reliable, as illustrated in Fig. 25.1. One way to cope with outliers is to remove them in advance, which will be discussed in Chapter 38. Another approach is to perform learning so that the solution is less sensitive to outliers. The ability to be less sensitive to outliers is referred to as *robustness*. In this chapter, robust learning methods are explored.

In the LS method, the goodness of fit to training samples is measured by the ℓ_2-loss function:

$$J_{\mathrm{LS}}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n} r_i^2,$$

An Introduction to Statistical Machine Learning. DOI: 10.1016/B978-0-12-802121-7.00036-4



(a) Without outliers (b) With an outlier

FIGURE 25.1

LS solution for straight-line model $f_{\boldsymbol{\theta}}(x) = \theta_1 + \theta_2 x$, which is strongly affected by an outlier.

where r_i denotes the *residual* for the ith training sample $(\boldsymbol{x}_i, y_i)$:

$$r_i = y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i).$$

Since the ℓ_2-loss squares the error, large error tends to be magnified significantly. For example, in Fig. 25.1 the output of the rightmost training sample is moved from $y \approx 3$ to $y \approx -4$. Then the residual 7 has the "power" of $7^2 = 49$ to pull the regression function down, which causes significant change in the solution.

25.2 ℓ_1-LOSS MINIMIZATION

To cope with the nonrobustness of the LS method, let us employ the ℓ_1-*loss* or the *absolute loss* (Fig. 25.2):

$$\widehat{\boldsymbol{\theta}}_{\mathrm{LA}} = \operatorname*{argmin}_{\boldsymbol{\theta}} J_{\mathrm{LA}}(\boldsymbol{\theta}), \quad \text{where } J_{\mathrm{LA}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} |r_i|.$$

This learning method is called ℓ_1-*loss minimization* or *least absolute deviations regression*.

An example of ℓ_1-loss minimization is shown in Fig. 25.3, which is obtained in the same setup as in Fig. 25.1 (how to compute the solution of least absolute deviations will be explained later in Section 25.3). This shows that ℓ_1-loss minimization is much more robust against outliers than ℓ_2-loss minimization, and ℓ_1-loss minimization gives almost the same solution as ℓ_2-loss minimization if there are no outliers.

For a constant model $f_\theta(\boldsymbol{x}) = \theta$, the LS method gives the *mean* of training output samples $\{y_i\}_{i=1}^n$:

$$\widehat{\theta}_{\mathrm{LS}} = \operatorname*{argmin}_{\theta} \sum_{i=1}^{n} (\theta - y_i)^2 = \mathrm{mean}\left(\{y_i\}_{i=1}^n\right).$$

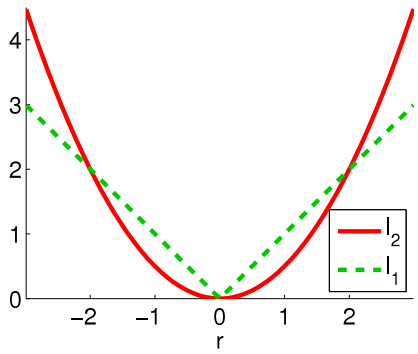
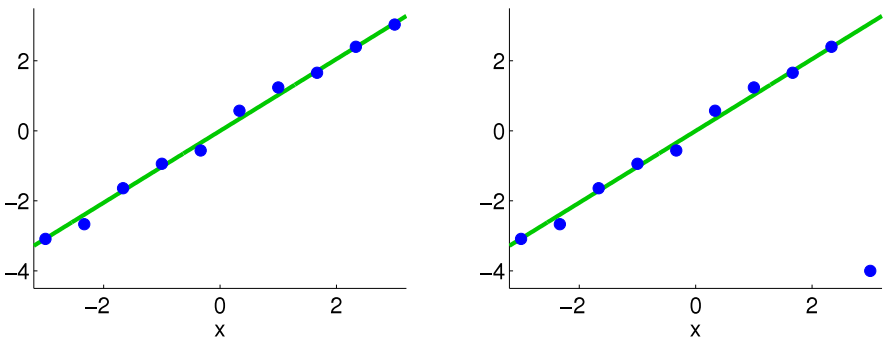


FIGURE 25.2

ℓ_2-loss and ℓ_1-loss. The ℓ_2-loss magnifies large residuals.



(a) Without outliers (b) With an outlier

FIGURE 25.3

Solution of least absolute deviations for straight-line model $f_{\boldsymbol{\theta}}(x) = \theta_1 + \theta_2 x$ for the same training samples as Fig. 25.1. Least absolute deviations give a much more robust solution than LS.

On the other hand, least absolute deviations gives the *median* (see Section 2.4.1) of the training output samples $\{y_i\}_{i=1}^n$:

$$\widehat{\theta}_{\mathrm{LA}} = \operatorname*{argmin}_{\theta} \sum_{i=1}^{n} |\theta - y_i| = \mathrm{median}\Big(\{y_i\}_{i=1}^n\Big).$$

The mean of $\{y_i\}_{i=1}^n$ is affected if one of the sample values is changed, while the median is not affected by changing sample values as long as the order of samples is

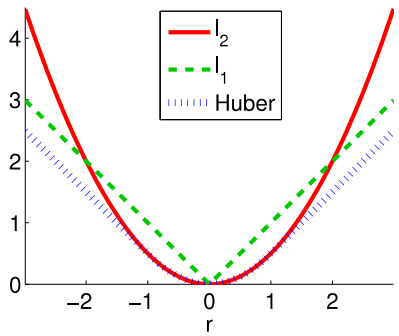


FIGURE 25.4

Huber loss, with threshold $\eta = 1$.

unchanged. The robustness of least absolute deviations is brought by such a property of the ℓ_1-loss.

25.3 HUBER LOSS MINIMIZATION

The ℓ_1-loss gives a robust solution, but high robustness implies that learning is performed without fully utilizing the information brought by training samples. In the extreme case, the estimator that does not learn anything from training samples but just always gives zero is the most robust approach, which is clearly meaningless. This extreme example implies that enhancing the robustness too much can degrade the performance of learning from inlier samples. The notion of *efficiency* discussed in Section 13.3 concerns the variance of an estimator, which can be interpreted as how much information can be learned from given training samples. Thus, achieving robustness and efficiency in a balanced way is crucial in practice. In this section, the *Huber loss* [58] is introduced, which can control the balance between robustness and efficiency.

25.3.1 DEFINITION

The Huber loss is defined with the ℓ_2-loss and the ℓ_1-loss as follows (Fig. 25.4):

$$\rho_{\mathrm{Huber}}(r) = \begin{cases} r^2/2 & (|r| \leq \eta), \\ \eta|r| - \eta^2/2 & (|r| > \eta). \end{cases}$$

This means that the Huber loss is reduced to the ℓ_2-loss if the absolute residual $|r|$ is less than or equal to a threshold η (i.e., that sample may be an inlier), while it is reduced to the ℓ_1-loss if the absolute residual $|r|$ is larger than a threshold η (i.e., that

sample may be an outlier). Note that the ℓ_1-loss is modified as $\eta|r| - \eta^2/2$ so that it is smoothly connected to the ℓ_2-loss $r^2/2$. The method of Huber loss minimization is given by

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \quad \text{where } J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \rho_{\text{Huber}}(r_i). \tag{25.1}$$

25.3.2 STOCHASTIC GRADIENT ALGORITHM

Let us consider the linear-in-parameter model:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b} \theta_j \phi_j(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}).$$

Since the Huber loss is once differentiable as

$$\rho'_{\text{Huber}}(r) = \begin{cases} r & (|r| \le \eta), \\ -\eta & (r < -\eta), \\ \eta & (r > \eta), \end{cases}$$

the solution of Huber loss minimization may be obtained by the *stochastic gradient* algorithm (see Section 15.3) as follows:

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \varepsilon \frac{\partial \rho_{\text{Huber}}}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}},$$

where $(\boldsymbol{x}_i, y_i)$ is a randomly chosen training sample and $\varepsilon > 0$ is the step size.

25.3.3 ITERATIVELY REWEIGHTED LS

Another approach is *iteratively reweighted LS*, which considers the following quadratic upper bound of the absolute-value part of the Huber loss:

$$\eta|r| - \frac{\eta^2}{2} \le \frac{\eta}{2c} r^2 + \frac{\eta c}{2} - \frac{\eta^2}{2} \quad \text{for } c > 0.$$

As illustrated in Fig. 25.5, this quadratic upper bound touches the Huber loss at $r = \pm c$.

Let us consider an iterative optimization procedure and construct an upper bound for $c_i = |\widetilde{r}_i|$:

$$\eta|r_i| - \frac{\eta^2}{2} \le \frac{\eta}{2|\widetilde{r}_i|} r_i^2 + \frac{\eta|\widetilde{r}_i|}{2} - \frac{\eta^2}{2},$$

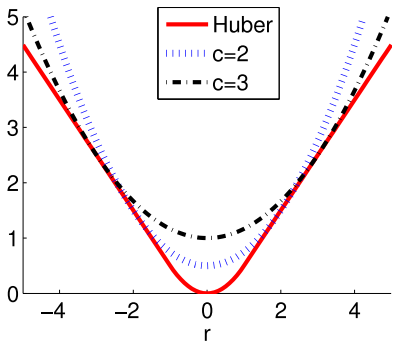


FIGURE 25.5

Quadratic upper bound $\frac{\eta r^2}{2c} + \frac{\eta c}{2} - \frac{\eta^2}{2}$ of Huber loss $\rho_{\text{Huber}}(r)$ for $c > 0$, which touches each other at $r = \pm c$.

where $\widetilde{r}_i$ denotes the residual of the current solution for the ith training sample $(\boldsymbol{x}_i, y_i)$. If the absolute-value part of the Huber loss is upper-bounded by this, Huber loss minimization problem Eq. (25.1) yields the following *weighted LS* problem:

$$\widehat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \widetilde{J}(\boldsymbol{\theta}), \quad \text{where } \widetilde{J}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n} \widetilde{w}_i r_i^2 + C,$$

where $C = \sum_{i:|\widetilde{r}_i|>\eta}(\eta|\widetilde{r}_i|/2 - \eta^2/2)$ is a constant that is independent of $\boldsymbol{\theta}$, and weight $\widetilde{w}_i$ is defined as follows (Fig. 25.6):

$$\widetilde{w}_i = \begin{cases} 1 & (|\widetilde{r}_i| \le \eta), \\ \eta/|\widetilde{r}_i| & (|\widetilde{r}_i| > \eta). \end{cases}$$

As explained in Section 22.1, the solution of weighted LS is given analytically as

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^\top \widetilde{\boldsymbol{W}} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \widetilde{\boldsymbol{W}} \boldsymbol{y}, \tag{25.2}$$

where $\widetilde{\boldsymbol{W}}$ denotes the diagonal matrix with diagonal elements $\widetilde{w}_1, \ldots, \widetilde{w}_n$.

Since the upper bound $\widetilde{J}$ touches the original objective function J at current solution $\boldsymbol{\theta} = \widetilde{\boldsymbol{\theta}}$,

$$J(\widetilde{\boldsymbol{\theta}}) = \widetilde{J}(\widetilde{\boldsymbol{\theta}})$$

holds. Since $\widehat{\boldsymbol{\theta}}$ is the minimizer of $\widetilde{J}$,

$$\widetilde{J}(\widetilde{\boldsymbol{\theta}}) \ge \widetilde{J}(\widehat{\boldsymbol{\theta}}).$$

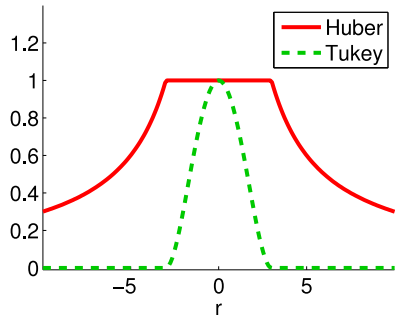


FIGURE 25.6

Weight functions for Huber loss minimization and Tukey loss minimization.

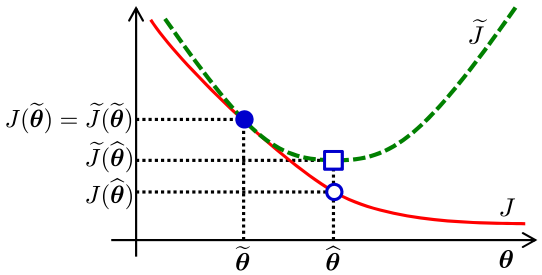


FIGURE 25.7

Updated solution $\widehat{\boldsymbol{\theta}}$ is no worse than current solution $\widetilde{\boldsymbol{\theta}}$.

Furthermore, $\widetilde{J}$ upper-bounds J and thus

$$\widetilde{J}(\widehat{\boldsymbol{\theta}}) \geq J(\widehat{\boldsymbol{\theta}})$$

holds. Summarizing the above relations,

$$J(\widetilde{\boldsymbol{\theta}}) = \widetilde{J}(\widetilde{\boldsymbol{\theta}}) \geq \widetilde{J}(\widehat{\boldsymbol{\theta}}) \geq J(\widehat{\boldsymbol{\theta}})$$

holds, meaning that the updated solution $\widehat{\boldsymbol{\theta}}$ is no worse than the current solution $\widetilde{\boldsymbol{\theta}}$ in terms of the objective value J (Fig. 25.7). By iterating this upper bound minimization, the solution of Huber loss minimization can be obtained. The pseudocode of this algorithm, called *iteratively reweighted LS*, is provided in Fig. 25.8. If *singular value*

1. Initialize parameter $\boldsymbol{\theta}$, e.g., by ordinary LS as

$$\boldsymbol{\theta} \longleftarrow (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y}.$$

2. Compute the weight matrix $\boldsymbol{W}$ from the current solution $\boldsymbol{\theta}$ as

$$\boldsymbol{W} = \mathrm{diag}\,(w_1, \ldots, w_n), \quad \text{where } w_i = \begin{cases} 1 & (|r_i| \leq \eta), \\ \eta / |r_i| & (|r_i| > \eta), \end{cases}$$

   and $r_i = y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ is the residual.

3. Compute the solution $\boldsymbol{\theta}$ based on the weight matrix $\boldsymbol{W}$ as

$$\boldsymbol{\theta} \longleftarrow (\boldsymbol{\Phi}^\top \boldsymbol{W} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{W} \boldsymbol{y}.$$

4. Iterate 2–3 until convergence.

FIGURE 25.8

Iteratively reweighted LS for Huber loss minimization.

decomposition (see Fig. 22.2) of design matrix,

$$\boldsymbol{\Phi} = \sum_{j=1}^{b} \kappa_j \boldsymbol{\psi}_j \boldsymbol{\varphi}_j^\top,$$

is computed in advance, update of $\boldsymbol{\theta}$ can be performed more efficiently as

$$\boldsymbol{\theta} \longleftarrow \sum_{j=1}^{b} \frac{\boldsymbol{\psi}_j^\top \boldsymbol{W} \boldsymbol{y}}{\kappa_j \boldsymbol{\psi}_j^\top \boldsymbol{W} \boldsymbol{\psi}_j} \boldsymbol{\varphi}_j.$$

A MATLAB code of iteratively reweighted LS for Huber loss minimization is provided in Fig. 25.9, and its behavior is illustrated in Fig. 25.10. This shows that only two iterations give almost the final solution. The iteration converges after four iterations and a robust solution can be obtained.

When the threshold η is taken to be sufficiently small, the Huber loss may be regarded as a smooth approximation to the ℓ_1-loss. Thus, the solution of ℓ_1-loss minimization can be approximately obtained by the above iteratively weighted LS algorithm.

25.3.4 ℓ_1-CONSTRAINED HUBER LOSS MINIMIZATION

As explained in Chapter 24, the ℓ_1-constraint tends to give a sparse solution. Here, ℓ_1-constrained Huber loss minimization is considered:

```
n=10; N=1000; x=linspace(-3,3,n)'; X=linspace(-4,4,N)';
y=x+0.2*randn(n,1); y(n)=-4;

p(:,1)=ones(n,1); p(:,2)=x; t=p\y; e=1;
for o=1:1000
  r=abs(p*t-y); w=ones(n,1); w(r>e)=e./r(r>e);
  t0=(p'*(repmat(w,1,2).*p))\(p'*(w.*y));
  if norm(t-t0)<0.001, break, end
  t=t0;
end
P(:,1)=ones(N,1); P(:,2)=X; F=P*t;
figure(1); clf; hold on; axis([-4 4 -4.5 3.5]);
plot(X,F,'g-'); plot(x,y,'bo');
```

FIGURE 25.9

MATLAB code of iteratively reweighted LS for Huber loss minimization. Straight-line model $f_{\boldsymbol{\theta}}(x) = \theta_1 + \theta_2 x$ is used, with threshold $\eta = 1$.

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \rho_{\text{Huber}}(r_i) \quad \text{subject to } \|\boldsymbol{\theta}\|_1 \le R^2,$$

which produces a robust and sparse solution.

As shown above, the optimization problem of Huber loss minimization can be solved by iteratively reweighted LS, which was obtained by a quadratic upper bounding technique. Similarly, ℓ_1-regularized LS, whose solution was obtained by ADMM in Section 24.2, can also be solved by iteratively reweighted LS. More specifically, the absolute value of parameter θ can be upper-bounded by a quadratic function as

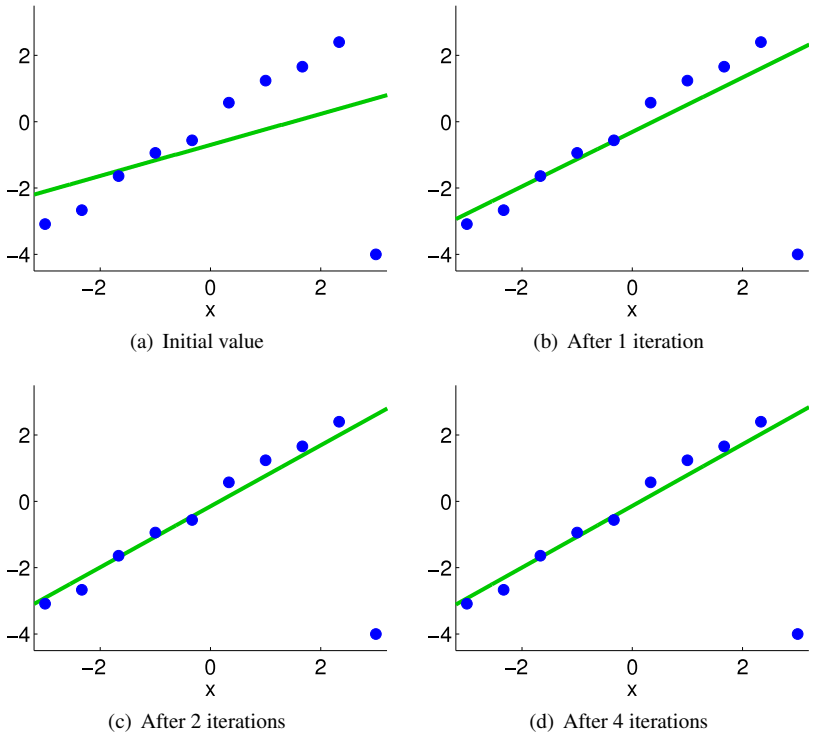
$$|\theta| \le \frac{\theta^2}{2c} + \frac{c}{2} \quad \text{for } c > 0,$$

which touches the absolute value at $\theta = \pm c$ (see Fig. 25.11).

Let us consider an iterative optimization procedure and construct an upper bound for $c_j = \widetilde{\theta}_j$:

$$|\theta_j| \le \frac{|\widetilde{\theta}_j|^\dagger}{2}\theta_j^2 + \frac{|\widetilde{\theta}_j|}{2},$$

where $\widetilde{\theta}_j$ is the current solution and † denotes the *generalized inverse* (see Fig. 22.1). If the ℓ_1-norm of parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_b)^\top$ is upper-bounded by this, the



(a) Initial value (b) After 1 iteration

(c) After 2 iterations (d) After 4 iterations

FIGURE 25.10

Examples of iteratively reweighted LS for Huber loss minimization. Straight-line model $f_{\boldsymbol{\theta}}(x) = \theta_1 + \theta_2 x$ is used, with threshold $\eta = 1$.

ℓ_1-regularized LS problem,

$$\min_{\boldsymbol{\theta}} \left[\frac{1}{2} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|_1 \right],$$

yields the following *generalized ℓ_2-regularized LS*:

$$\widehat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \widetilde{J}(\boldsymbol{\theta}), \quad \text{where } \widetilde{J}(\boldsymbol{\theta}) = J_{\mathrm{LS}}(\boldsymbol{\theta}) + \frac{\lambda}{2} \boldsymbol{\theta}^\top \widetilde{\boldsymbol{\Theta}}^\dagger \boldsymbol{\theta} + C,$$

where $C = \sum_{j=1}^{b} |\widetilde{\theta}_j|/2$ is a constant that is independent of $\boldsymbol{\theta}$. As explained in Section 23.2, the solution of generalized ℓ_2-regularized LS is given analytically as

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \widetilde{\boldsymbol{\Theta}}^\dagger)^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y}.$$

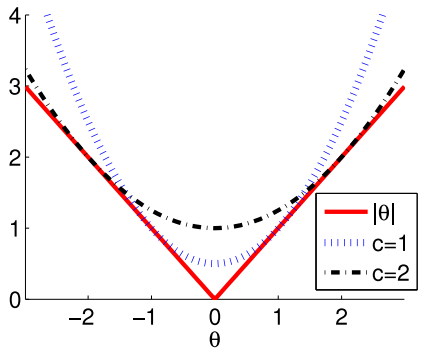


FIGURE 25.11

Quadratic upper bound $\frac{\theta^2}{2c} + \frac{c}{2}$ of absolute value $|\theta|$ for $c > 0$, which touches each other at $\theta = \pm c$.

1. Initialize parameter $\boldsymbol{\theta}$, e.g., randomly or by ordinary LS as

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\Phi}^{\dagger} \boldsymbol{y}.$$

2. Compute the weight matrix $\boldsymbol{W}$ and regularization matrix $\boldsymbol{\Theta}$ from the current solution $\boldsymbol{\theta}$ as

$$\boldsymbol{W} = \mathrm{diag}\,(w_1, \ldots, w_n) \quad \text{and} \quad \boldsymbol{\Theta} = \mathrm{diag}\,(|\theta_1|, \ldots, |\theta_b|)\,,$$

where weight w_i is defined using residual $r_i = y_i - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ as

$$w_i = \begin{cases} 1 & (|r_i| \leq \eta), \\ \eta / |r_i| & (|r_i| > \eta). \end{cases}$$

3. Compute the solution $\boldsymbol{\theta}$ based on the weight matrix $\boldsymbol{W}$ regularization matrix $\boldsymbol{\Theta}$ as

$$\boldsymbol{\theta} \longleftarrow (\boldsymbol{\Phi}^{\top} \boldsymbol{W} \boldsymbol{\Phi} + \lambda \boldsymbol{\Theta}^{\dagger})^{-1} \boldsymbol{\Phi}^{\top} \boldsymbol{W} \boldsymbol{y}.$$

4. Iterate 2–3 until convergence.

FIGURE 25.12

Iteratively reweighted LS for ℓ_1-regularized Huber loss minimization.

```
n=50; x=linspace(-3,3,n)'; pix=pi*x;
y=sin(pix)./(pix)+0.1*x+0.2*randn(n,1); y(n/2)=-0.5;
hh=2*0.3^2; l=0.1; e=0.1; t=randn(n,1); x2=x.^2;
k=exp(-(repmat(x2,1,n)+repmat(x2',n,1)-2*x*x')/hh);
for o=1:1000
  r=abs(k*t-y); w=ones(n,1); w(r>e)=e./r(r>e);
  Z=k*(repmat(w,1,n).*k)+l*pinv(diag(abs(t)));
  t0=(Z+0.000001*eye(n))\(k*(w.*y));
  if norm(t-t0)<0.001, break, end
  t=t0;
end
N=1000; X=linspace(-3,3,N)';
K=exp(-(repmat(X.^2,1,n)+repmat(x2',N,1)-2*X*x')/hh); F=K*t;
figure(1); clf; hold on; axis([-2.8 2.8 -1 1.5]);
plot(X,F,'g-'); plot(x,y,'bo');
```

FIGURE 25.13

MATLAB code of iteratively reweighted LS for ℓ_1-regularized Huber loss minimization with Gaussian kernel model.

The iteratively reweighted LS algorithms for ℓ_1-regularized LS and Huber loss minimization can be combined to obtain a sparse and robust solution, as summarized in Fig. 25.12.

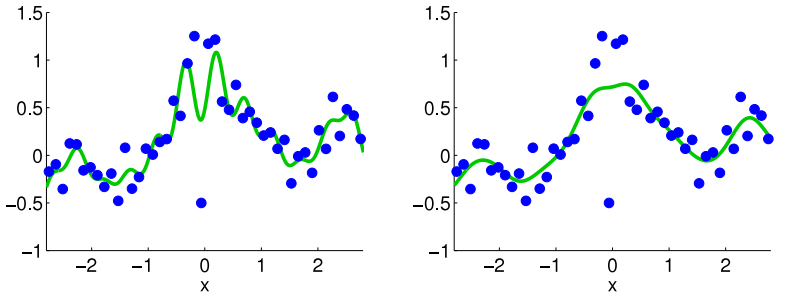
A MATLAB code of iteratively reweighted LS for ℓ_1-regularized Huber loss minimization is provided in Fig. 25.13, where the Gaussian kernel model,

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_j\|^2}{2h^2}\right),$$

is used and the Gaussian bandwidth is set at $h = 0.3$. Since plain LS is numerically unstable, it is stabilized by adding 10^{-6} to the diagonal elements of $\boldsymbol{\Phi}^\top \boldsymbol{W} \boldsymbol{\Phi}$ (more precisely, $\boldsymbol{KWK}$ since the Gaussian kernel model is used). The behavior of ℓ_1-regularized Huber loss minimization is illustrated in Fig. 25.14. Regardless of the presence or absence of the ℓ_1-regularizer, ℓ_2-loss minimization is strongly affected by the outlier at around $x = 0$, while Huber loss minimization can successfully suppress its influence. With the ℓ_1-regularizer, ℓ_2-loss minimization gives 38 zero parameters out of 50 parameters, and Huber loss minimization gives 36 zero parameters.

25.4 TUKEY LOSS MINIMIZATION

The Huber loss nicely controls the balance between efficiency and robustness by combining the ℓ_1-loss and the ℓ_2-loss. However, as long as the ℓ_1-loss is used, strong



$(\lambda, \eta) = (0, \infty)$
(a) ℓ_2-loss minimization

$(\lambda, \eta) = (0.1, \infty)$
(b) ℓ_1-regularized ℓ_2-loss minimization

$(\lambda, \eta) = (0, 0.1)$
(c) Huber loss minimization

$(\lambda, \eta) = (0.1, 0.1)$
(d) ℓ_1-regularized Huber loss minimization

FIGURE 25.14

Example of ℓ_1-regularized Huber loss minimization with Gaussian kernel model.

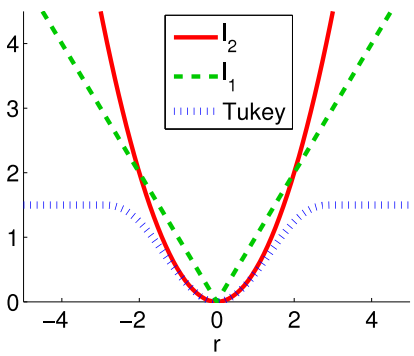
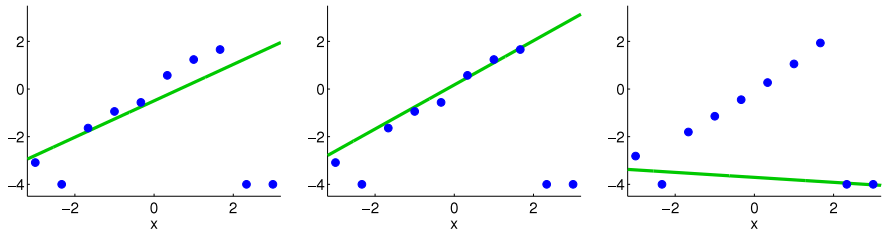


FIGURE 25.15

Tukey loss, with threshold $\eta = 3$.



(a) Huber loss minimization (b) Tukey loss minimization 1 (c) Tukey loss minimization 2

FIGURE 25.16

Example of Tukey loss minimization. Tukey loss minimization gives more robust solutions than Huber loss minimization, but only a local optimal solution can be obtained.

outliers can still affect the solution significantly. Indeed, as illustrated in Fig. 25.6, the weight $\widetilde{w}_i$ used in Huber loss minimization is not zero for large absolute residuals.

In such a hard circumstance, the *Tukey loss* [73] is useful (Fig. 25.15):

$$\rho_{\text{Tukey}}(r) = \begin{cases} \left(1 - \left[1 - r^2/\eta^2\right]^3\right)\eta^2/6 & (|r| \le \eta), \\ \eta^2/6 & (|r| > \eta). \end{cases}$$

The Tukey loss is constant $\eta^2/6$ if the absolute residual $|r|$ is larger than a threshold η. Thus, Tukey loss minimization is expected to be more robust than Huber loss minimization.

However, the Tukey loss is not a *convex function* (see Fig. 8.3). Thus, there may exist multiple local optimal solutions and finding the global one is not straightforward. In practice, *iteratively reweighted LS* with the following weight is used to find a local optimal solution (Fig. 25.6):

$$w = \begin{cases} \left(1 - r^2/\eta^2\right)^2 & (|r| \le \eta), \\ 0 & (|r| > \eta). \end{cases}$$

Since the Tukey weight is zero for $|r| > \eta$, Tukey loss minimization is not at all influenced by strong outliers.

A MATLAB code of iteratively reweighted LS for Tukey loss minimization is essentially the same as that for Huber loss minimization shown in Fig. 25.9. Only the weight computation for the Huber loss,

```
w=ones(n,1); w(r>e)=e./r(r>e);
```

is replaced with that for the Tukey loss:

```
w=zeros(n,1); w(r<=e)=(1-r(r<=e).^2/e^2).^2;
```

The behavior of Tukey loss minimization is illustrated in Fig. 25.16, where the threshold is set at $\eta = 1$. In this example, Tukey loss minimization gives an even more robust solution than Huber loss minimization. However, since Tukey loss minimization is a nonconvex optimization problem, different solutions may be obtained if the initial solution is changed or training samples are slightly perturbed. Indeed, as illustrated in Fig. 25.16(c), slightly changing the noise included in training output samples gives another local optimal solution, which is significantly different from the original one in Fig. 25.16(b).

For general differentiable symmetric loss $\rho(r)$, iteratively reweighted LS with weight $w_i = \rho'(r_i)/r_i$ gives a (local) optimal solution, where $\rho'(r)$ denotes the derivative of $\rho(r)$ [57].