Pattern Information Processing: Robust Method

Masashi Sugiyama
(Department of Computer Science)

Contact: W8E-505
sugi@cs.titech.ac.jp
http://sugiyama-www.cs.titech.ac.jp/~sugi/

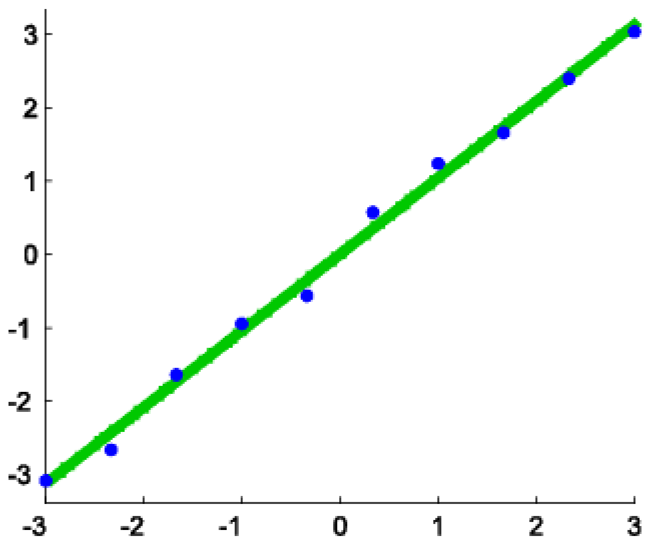
Outliers

- In practice, very large noise sometimes appears.
- Furthermore, irregular values can be observed by measurement trouble or by human error.
- Samples with such irregular values are called outliers.

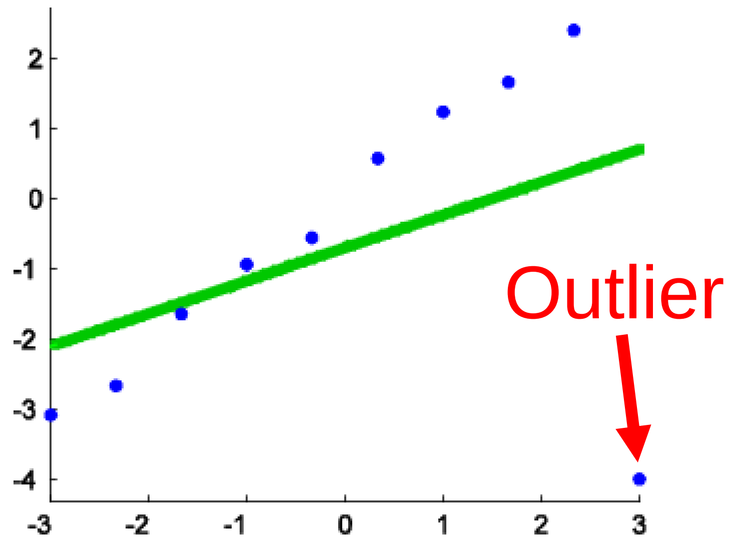
Outliers (cont.)

- LS criterion is sensitive to outliers.

$$f_{\boldsymbol{\alpha}}(x) = \alpha_1 + \alpha_2 x$$



LS (without outlier)



LS (with outlier)

- Even a single outlier can corrupt the learning result!

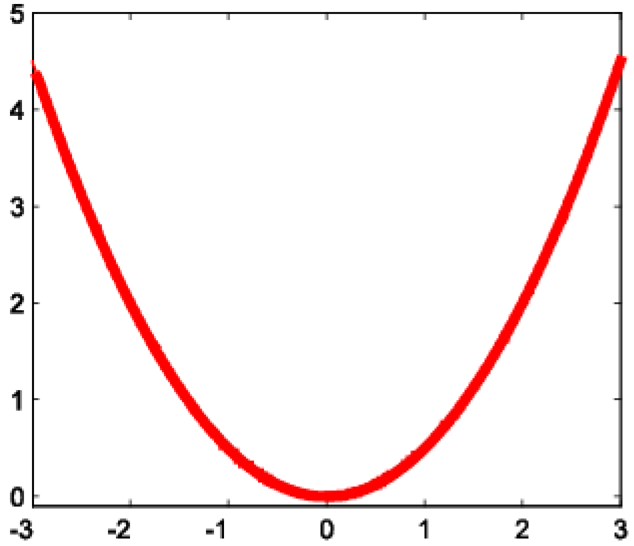
Today’s Plan

- Robust learning with ℓ_1 -loss
- Robustness and convexity
- Robustness and efficiency
- Robust learning with Huber’s loss
- Robustness and sparsity

Quadratic Loss

$$J_{LS}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \left(f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) - y_i \right)^2$$

- In LS, goodness-of-fit is measured by the squared loss.
- Therefore, even a single outlier has quadratic power to "pull" the learned function.
- The solution will be robust if outliers are deemphasized.

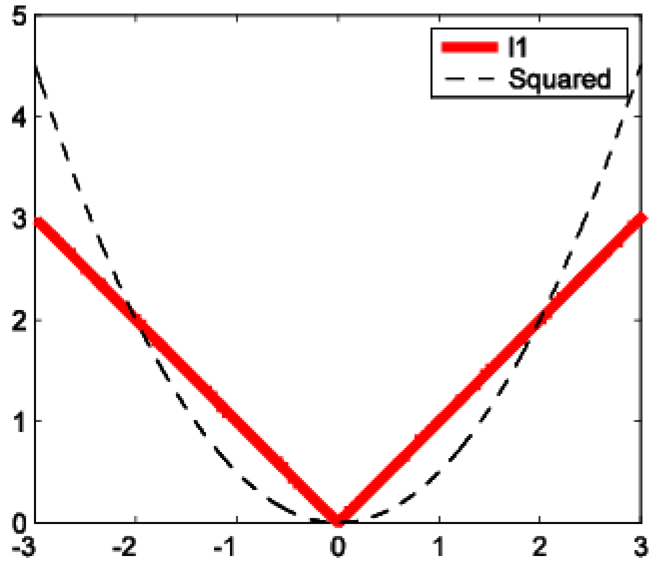


l1-Loss

- Use ℓ_1-loss for measuring goodness-of-fit:

$$\hat{\boldsymbol{\alpha}}_{\ell_1} = \mathop{\mathrm{argmin}}_{\boldsymbol{\alpha} \in \mathbb{R}^b} \left[\sum_{i=1}^{n} \left| f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) - y_i \right| \right]$$

- Outliers influence only linearly!



How to Obtain a Solution

$$\hat{\boldsymbol{\alpha}}_{\ell_1} = \mathop{\mathrm{argmin}}_{\boldsymbol{\alpha}\in\mathbb{R}^b} \left[\sum_{i=1}^{n} \left| f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) - y_i \right| \right] \qquad f_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{i=1}^{b} \alpha_i \varphi_i(\boldsymbol{x})$$

- Use the ℓ_1-trick:

$$|\epsilon| = \min_{v\in\mathbb{R}} v \quad \text{subject to } -v \leq \epsilon \leq v$$

- $\hat{\boldsymbol{\alpha}}_{\ell_1}$ is given as the solution of the following linearly-constrained linear program:

$$\mathop{\mathrm{argmin}}_{\boldsymbol{\alpha}\in\mathbb{R}^b, \boldsymbol{v}\in\mathbb{R}^n} \left[\sum_{i=1}^{n} v_i \right]$$

$$\text{subject to } -\boldsymbol{v} \leq \boldsymbol{X}\boldsymbol{\alpha} - \boldsymbol{y} \leq \boldsymbol{v}$$

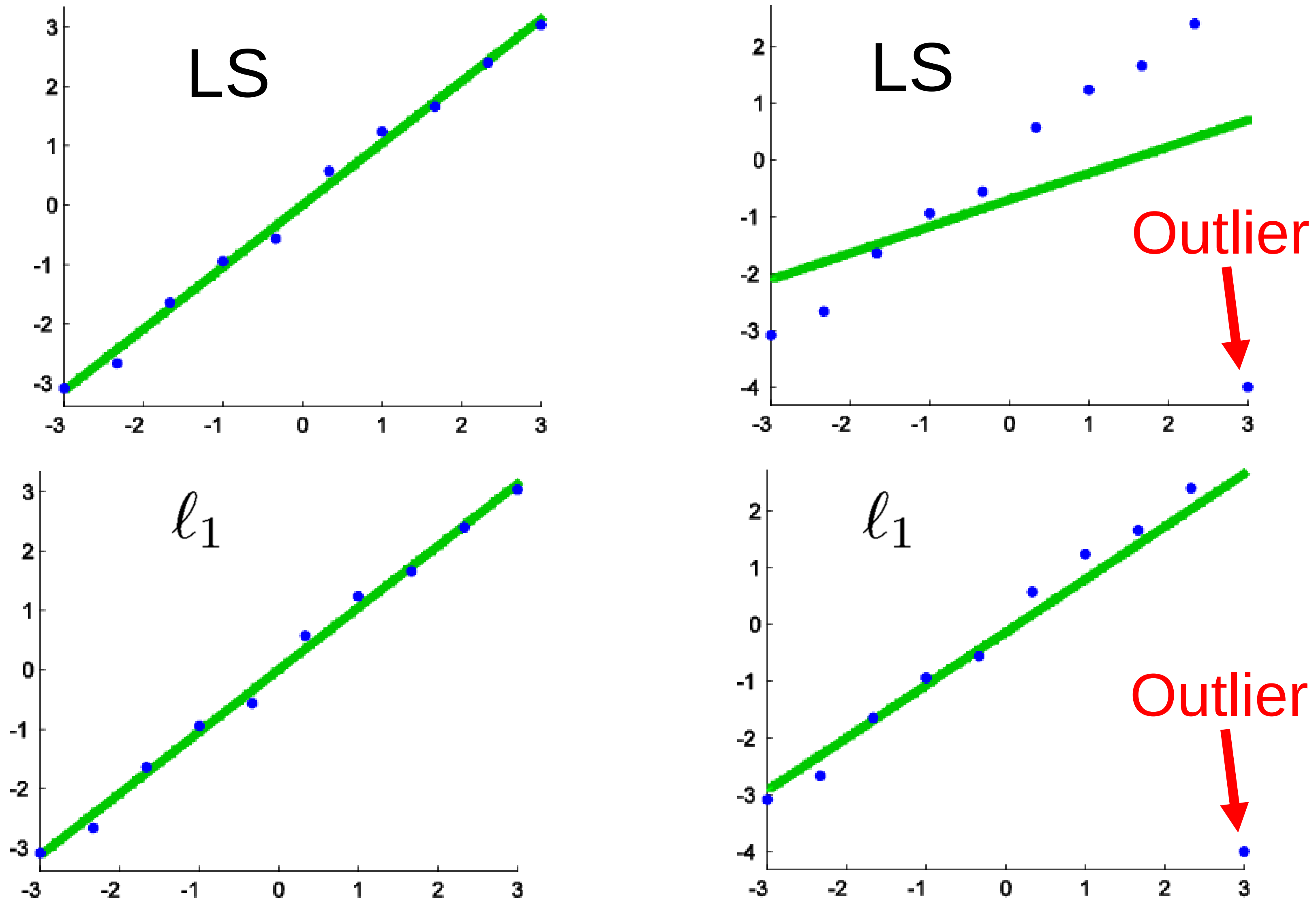
Linearly-Constrained Linear Program (LP)

- Standard optimization software can solve LP:

$$\min_{\boldsymbol{\beta}} \langle \boldsymbol{\beta}, \boldsymbol{q} \rangle \quad \text{subject to } \boldsymbol{H}\boldsymbol{\beta} \leq \boldsymbol{h}$$
$$\boldsymbol{G}\boldsymbol{\beta} = \boldsymbol{g}$$

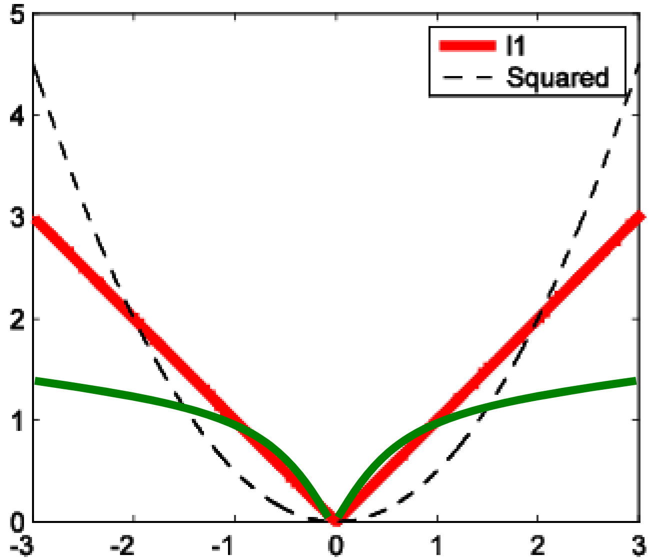
- Let $\boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{v} \end{pmatrix}$ $\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} = (\boldsymbol{I}_b, \boldsymbol{O}_{b \times n})$ $\boldsymbol{\Gamma}_{\boldsymbol{v}} = (\boldsymbol{O}_{n \times b}, \boldsymbol{I}_n)$ $\Rightarrow$ $\boldsymbol{\alpha} = \boldsymbol{\Gamma}_{\boldsymbol{\alpha}}\boldsymbol{\beta}$ $\boldsymbol{v} = \boldsymbol{\Gamma}_{\boldsymbol{v}}\boldsymbol{\beta}$
  - $\displaystyle\sum_{i=1}^{n} v_i$ $\Rightarrow$ $\langle \boldsymbol{\beta}, \boldsymbol{\Gamma}_{\boldsymbol{v}}^{\top} \boldsymbol{1}_n \rangle$
  - $-\boldsymbol{v} \leq \boldsymbol{X}\boldsymbol{\alpha} - \boldsymbol{y} \leq \boldsymbol{v}$ $\Rightarrow$ $\begin{pmatrix} -\boldsymbol{X}\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} - \boldsymbol{\Gamma}_{\boldsymbol{v}} \\ \boldsymbol{X}\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} - \boldsymbol{\Gamma}_{\boldsymbol{v}} \end{pmatrix} \boldsymbol{\beta} \leq \begin{pmatrix} -\boldsymbol{y} \\ \boldsymbol{y} \end{pmatrix}$

Examples



Robustness and Convexity

- Influence of outliers can be further reduced by using a sub-linear loss:



- However, such a sub-linear loss is non-convex.
- Obtaining a global optimal solution is difficult.

Statistical Interpretation

- Data: Observation = True value + Noise

$$\{y_i \mid y_i = \mu^* + \epsilon_i\}_{i=1}^n$$

- Goal: Estimate μ^* from $\{y_i\}_{i=1}^n$.
- ℓ_2-loss: Sample mean is the solution.

$$\widehat{\mu}_{\ell_2} = \operatorname*{argmin}_{\mu} \left[\sum_{i=1}^n (y_i - \mu)^2 \right] = \mathrm{mean}\left(\{y_i\}_{i=1}^n\right)$$

- ℓ_1-loss: Sample median is the solution.

$$\widehat{\mu}_{\ell_1} = \operatorname*{argmin}_{\mu} \left[\sum_{i=1}^n |y_i - \mu| \right] = \mathrm{median}\left(\{y_i\}_{i=1}^n\right)$$

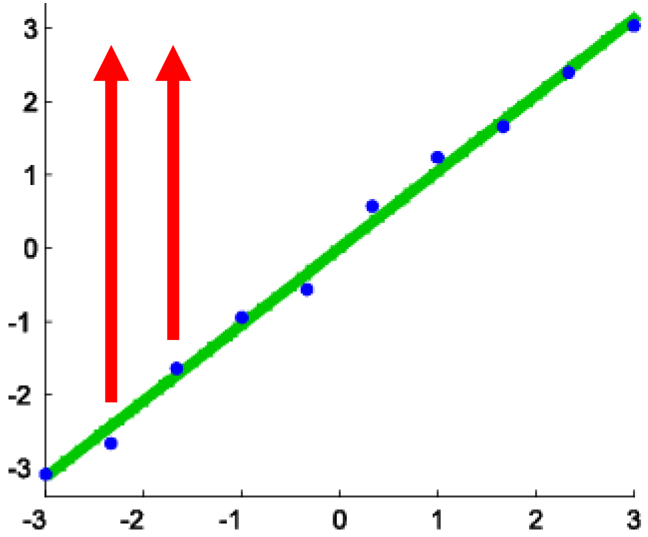
Proof: Homework!

Robustness and Efficiency

- We move α% of samples to infinity.
- Breakdown point: The maximum α with which a learned function still stays finite.
  - ℓ_2-loss: 0%

    ➡ Not at all robust
  - ℓ_1-loss: 50%

    ➡ Most robust



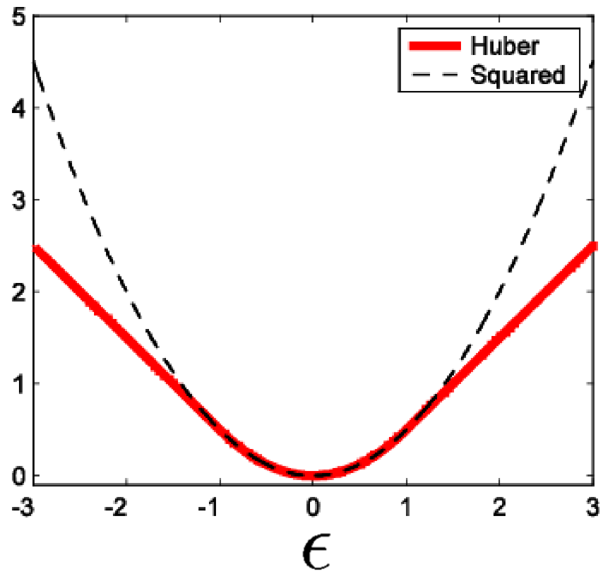
- However, ℓ_1-loss is not statistically efficient for Gaussian noise (i.e., having larger variance)

Huber's Robust Learning

$$\hat{\boldsymbol{\alpha}}_{Huber} = \mathop{\mathrm{argmin}}_{\boldsymbol{\alpha}\in\mathbb{R}^b} \sum_{i=1}^{n} \rho\Big(f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) - y_i\Big)$$

$$\rho(\epsilon) = \begin{cases} \frac{1}{2}\epsilon^2 & (|\epsilon| \leq t) \\ t|\epsilon| - \frac{1}{2}t^2 & (|\epsilon| > t) \end{cases}$$

$t \geq 0$



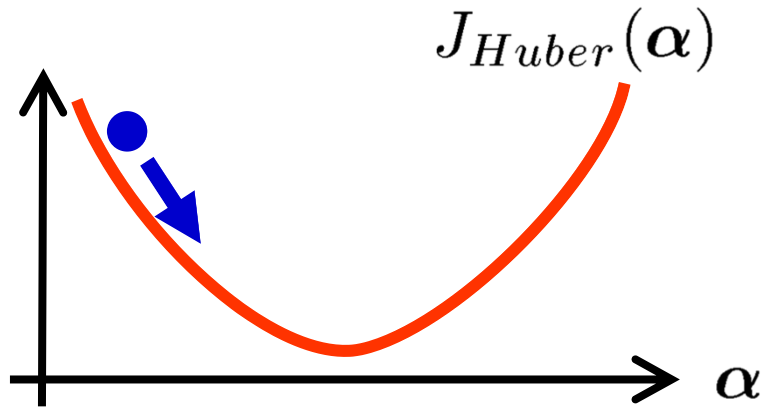
- ℓ_2-loss for inliers (samples with small errors).
- ℓ_1-loss for outliers (samples with large errors).

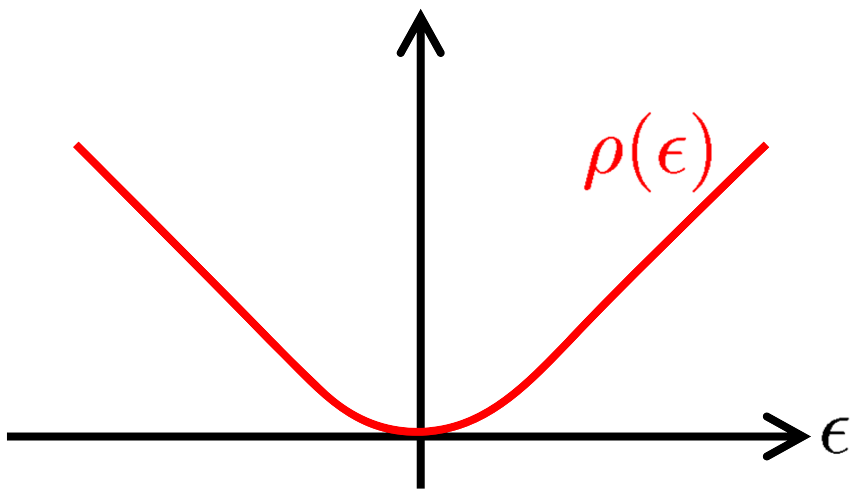
P. J. Huber, Robust Statistics, Wiley, 1981.

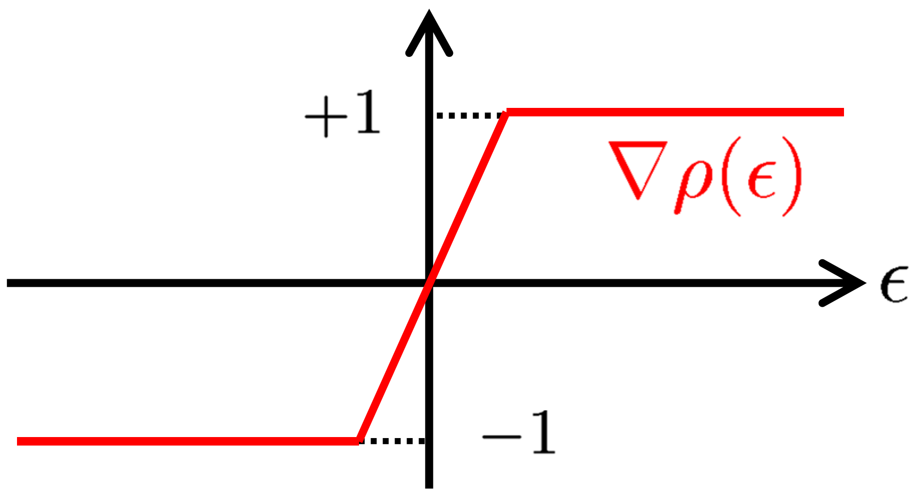
How to Obtain A Solution: Gradient Descent

$$\boldsymbol{\alpha} \longleftarrow \boldsymbol{\alpha} - \epsilon \nabla J_{Huber}(\boldsymbol{\alpha})$$

$$J_{Huber}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \rho\Big(f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) - y_i\Big)$$







- A quasi-Newton method may also be used.

Quadratic Program (QP)

- Another expression of Huber's loss:

$$\rho(y) = \min_{v \in \mathbb{R}} g(v) \qquad g(v) = \frac{1}{2}v^2 + t|y - v|$$

- Then $\hat{\boldsymbol{\alpha}}_{Huber}$ can be obtained as the solution of

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^b, \boldsymbol{v} \in \mathbb{R}^n} \left[\frac{1}{2} \|\boldsymbol{v}\|^2 + t \|\boldsymbol{X}\boldsymbol{\alpha} - \boldsymbol{y} - \boldsymbol{v}\|_1 \right]$$

- Using the ℓ_1-trick, this is expressed as QP:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^b, \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n} \left[\frac{1}{2} \|\boldsymbol{v}\|^2 + t \sum_{i=1}^{n} u_i \right]$$

$$\text{subject to } -\boldsymbol{u} \leq \boldsymbol{X}\boldsymbol{\alpha} - \boldsymbol{y} - \boldsymbol{v} \leq \boldsymbol{u}$$

Transforming into Standard Form

$$\min_{\boldsymbol{\beta}} \left[\frac{1}{2} \langle \boldsymbol{Q}\boldsymbol{\beta}, \boldsymbol{\beta} \rangle + \langle \boldsymbol{\beta}, \boldsymbol{q} \rangle \right] \quad \text{subject to } \boldsymbol{H}\boldsymbol{\beta} \leq \boldsymbol{h}, \quad \boldsymbol{G}\boldsymbol{\beta} = \boldsymbol{g}$$

- Let $\boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix}$

$$\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} = (\boldsymbol{I}_b, \boldsymbol{O}_{b\times n}, \boldsymbol{O}_{b\times n}), \quad \boldsymbol{\Gamma}_{\boldsymbol{u}} = (\boldsymbol{O}_{n\times b}, \boldsymbol{I}_n, \boldsymbol{O}_{n\times n}), \quad \boldsymbol{\Gamma}_{\boldsymbol{v}} = (\boldsymbol{O}_{n\times b}, \boldsymbol{O}_{n\times n}, \boldsymbol{I}_n)$$

$$\Rightarrow \quad \boldsymbol{\alpha} = \boldsymbol{\Gamma}_{\boldsymbol{\alpha}}\boldsymbol{\beta}, \quad \boldsymbol{u} = \boldsymbol{\Gamma}_{\boldsymbol{u}}\boldsymbol{\beta}, \quad \boldsymbol{v} = \boldsymbol{\Gamma}_{\boldsymbol{v}}\boldsymbol{\beta}$$

- $\frac{1}{2}\|\boldsymbol{v}\|^2 + t\sum_{i=1}^{n} u_i \quad \Rightarrow \quad \frac{1}{2}\langle \boldsymbol{\Gamma}_{\boldsymbol{v}}^\top \boldsymbol{\Gamma}_{\boldsymbol{v}}\boldsymbol{\beta}, \boldsymbol{\beta} \rangle + \langle \boldsymbol{\beta}, t\boldsymbol{\Gamma}_{\boldsymbol{u}}^\top \boldsymbol{1}_n \rangle$

- $-\boldsymbol{u} \leq \boldsymbol{X}\boldsymbol{\alpha} - \boldsymbol{y} - \boldsymbol{v} \leq \boldsymbol{u}$

$$\Rightarrow \quad \begin{pmatrix} -\boldsymbol{X}\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} - \boldsymbol{\Gamma}_{\boldsymbol{u}} + \boldsymbol{\Gamma}_{\boldsymbol{v}} \\ \boldsymbol{X}\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} - \boldsymbol{\Gamma}_{\boldsymbol{u}} - \boldsymbol{\Gamma}_{\boldsymbol{v}} \end{pmatrix} \boldsymbol{\beta} \leq \begin{pmatrix} -\boldsymbol{y} \\ \boldsymbol{y} \end{pmatrix}$$

Robustness and Sparseness

- Huber's method does not generally provide a sparse solution.
- Combining Huber's loss with ℓ_1-penalty:

$$\hat{\boldsymbol{\alpha}}_{SparseHuber} = \mathop{\mathrm{argmin}}_{\boldsymbol{\alpha}\in\mathbb{R}^b} \left[\sum_{i=1}^{n} \rho\Big(f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) - y_i \Big) + \lambda \|\boldsymbol{\alpha}\|_1 \right]$$

- An approximate solution $\hat{\boldsymbol{\alpha}}_{SparseHuber}$ can be obtained by approximate gradient descent.

Linear Programming Learning

- Combine ℓ_1 -loss and ℓ_1-constraint:

$$\hat{\boldsymbol{\alpha}}_{\ell_1} = \mathop{\mathrm{argmin}}_{\boldsymbol{\alpha}\in\mathbb{R}^b} \left[\sum_{i=1}^{n} \left| f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) - y_i \right| + \lambda \|\boldsymbol{\alpha}\|_1 \right]$$

- Using the ℓ_1 -trick, we can obtain $\hat{\boldsymbol{\alpha}}_{LP}$ as the solution of the following LP:

$$\mathop{\mathrm{argmin}}_{\boldsymbol{\alpha},\boldsymbol{u}\in\mathbb{R}^b,\boldsymbol{v}\in\mathbb{R}^n} \left[\sum_{i=1}^{n} v_i + \lambda \sum_{i=1}^{b} u_i \right]$$

$$\text{subject to } -\boldsymbol{v} \leq \boldsymbol{X}\boldsymbol{\alpha} - \boldsymbol{y} \leq \boldsymbol{v}$$

$$-\boldsymbol{u} \leq \boldsymbol{\alpha} \leq \boldsymbol{u}$$

Transforming into Standard Form

$$\min_{\boldsymbol{\beta}} \langle \boldsymbol{\beta}, \boldsymbol{q} \rangle \quad \text{subject to } \boldsymbol{H}\boldsymbol{\beta} \leq \boldsymbol{h}$$
$$\boldsymbol{G}\boldsymbol{\beta} = \boldsymbol{g}$$

- Let $\boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix}$ $\quad \boldsymbol{\Gamma}_{\boldsymbol{\alpha}} = (\boldsymbol{I}_b, \boldsymbol{O}_{b\times b}, \boldsymbol{O}_{b\times n})$, $\boldsymbol{\Gamma}_{\boldsymbol{u}} = (\boldsymbol{O}_{b\times b}, \boldsymbol{I}_b, \boldsymbol{O}_{b\times n})$, $\boldsymbol{\Gamma}_{\boldsymbol{v}} = (\boldsymbol{O}_{n\times b}, \boldsymbol{O}_{n\times b}, \boldsymbol{I}_n)$ $\Rightarrow$ $\boldsymbol{\alpha} = \boldsymbol{\Gamma}_{\boldsymbol{\alpha}}\boldsymbol{\beta}$, $\boldsymbol{u} = \boldsymbol{\Gamma}_{\boldsymbol{u}}\boldsymbol{\beta}$, $\boldsymbol{v} = \boldsymbol{\Gamma}_{\boldsymbol{v}}\boldsymbol{\beta}$
- $\displaystyle\sum_{i=1}^{n} v_i + \lambda \sum_{i=1}^{b} u_i$ $\Rightarrow$ $\langle \boldsymbol{\beta}, \boldsymbol{\Gamma}_{\boldsymbol{v}}^\top \mathbf{1}_n + \lambda \boldsymbol{\Gamma}_{\boldsymbol{u}}^\top \mathbf{1}_b \rangle$
- $-\boldsymbol{v} \leq \boldsymbol{X}\boldsymbol{\alpha} - \boldsymbol{y} \leq \boldsymbol{v}$, $\quad -\boldsymbol{u} \leq \boldsymbol{\alpha} \leq \boldsymbol{u}$ $\Rightarrow$ $\begin{pmatrix} -\boldsymbol{X}\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} - \boldsymbol{\Gamma}_{\boldsymbol{v}} \\ \boldsymbol{X}\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} - \boldsymbol{\Gamma}_{\boldsymbol{v}} \\ -\boldsymbol{\Gamma}_{\boldsymbol{\alpha}} - \boldsymbol{\Gamma}_{\boldsymbol{u}} \\ \boldsymbol{\Gamma}_{\boldsymbol{\alpha}} - \boldsymbol{\Gamma}_{\boldsymbol{u}} \end{pmatrix} \boldsymbol{\beta} \leq \begin{pmatrix} -\boldsymbol{y} \\ \boldsymbol{y} \\ \mathbf{0}_b \\ \mathbf{0}_b \end{pmatrix}$

Combinations of Various Losses and Penalties

Loss \ Penalty	None	ℓ_2 Smooth	ℓ_1 Smooth & Sparse
ℓ_2 -loss Efficient	Analytic	Analytic	QP, AGD
Huber	QP, GD	QP, GD	QP, AGD
ℓ_1 -loss Robust	LP, AGD	QP, AGD	LP, AGD

- QP: Quadratic Program, LP: Linear Program, GD: Gradient Descent, AGD: Approximate GD.

Homework

1. Prove

$$\widehat{\mu}_{\ell_2} = \underset{\mu}{\operatorname{argmin}} \left[\sum_{i=1}^{n} (y_i - \mu)^2 \right] = \operatorname{mean}\left(\{y_i\}_{i=1}^n\right)$$

$$\widehat{\mu}_{\ell_1} = \underset{\mu}{\operatorname{argmin}} \left[\sum_{i=1}^{n} |y_i - \mu| \right] = \operatorname{median}\left(\{y_i\}_{i=1}^n\right)$$

under $\{y_i \mid y_i = \mu^* + \epsilon_i\}_{i=1}^n$.

Homework (cont.)

2. For your own toy 1-dimensional data, perform simulations using
   - Linear/Gaussian kernel models
   - Huber learning

   and analyze the results, e.g., by changing
   - Target functions
   - Number of samples
   - Noise level

   Including outliers in the dataset would be essential for this homework.