April 22, 2025

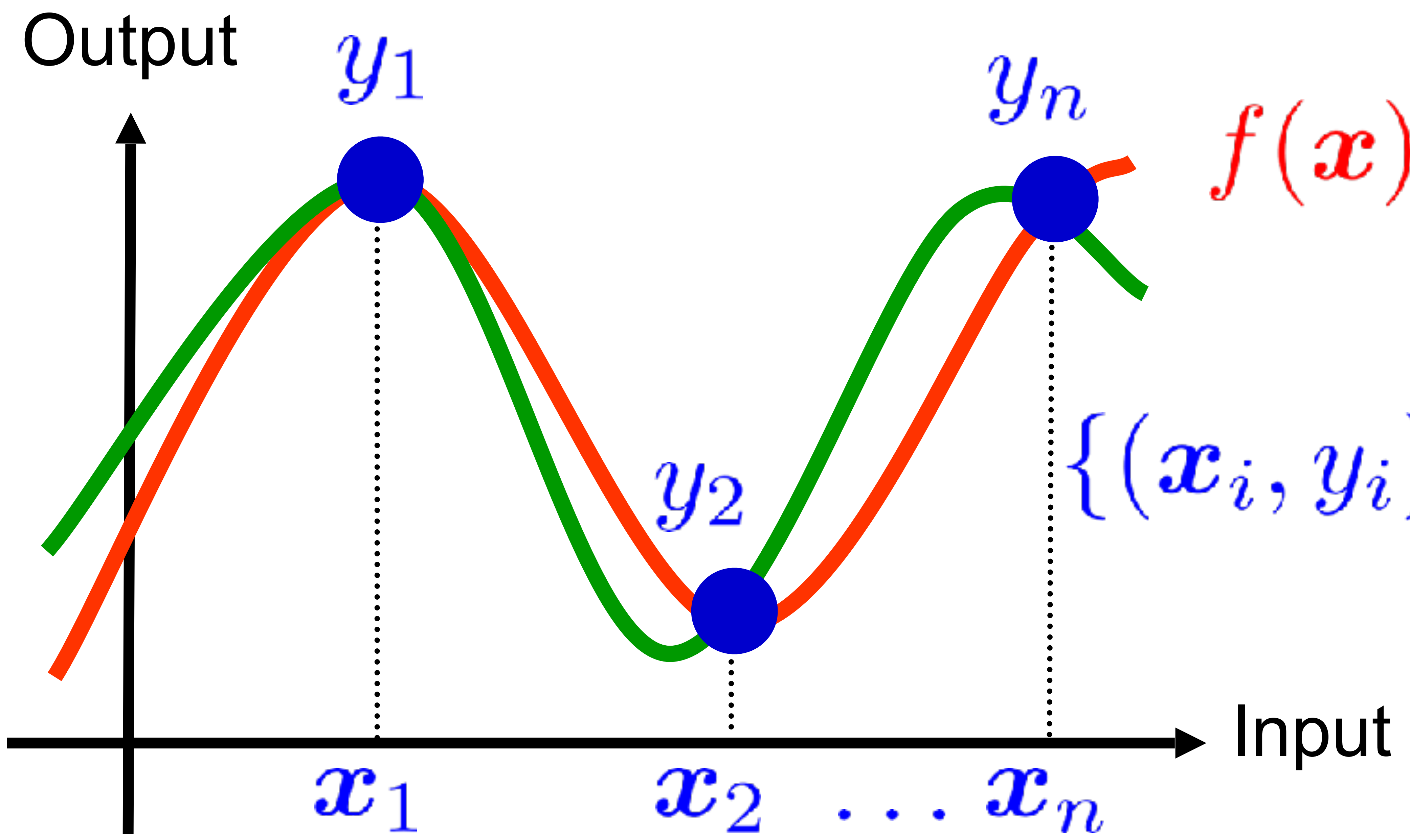
Regression 2

Masashi Sugiyama, Takashi Ishida

sugi@k.u-tokyo.ac.jp, ishi@k.u-tokyo.ac.jp

http://www.ms.k.u-tokyo.ac.jp

Regression = Function Approximation



$f(\boldsymbol{x})$: function to be learned

$\hat{f}(\boldsymbol{x})$: learned function

$\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$: training samples

$$y_i = f(\boldsymbol{x}_i) \ (+\text{noise})$$

Learn a function that is close to the underlying function w/ training samples

Linear-in-parameter model

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

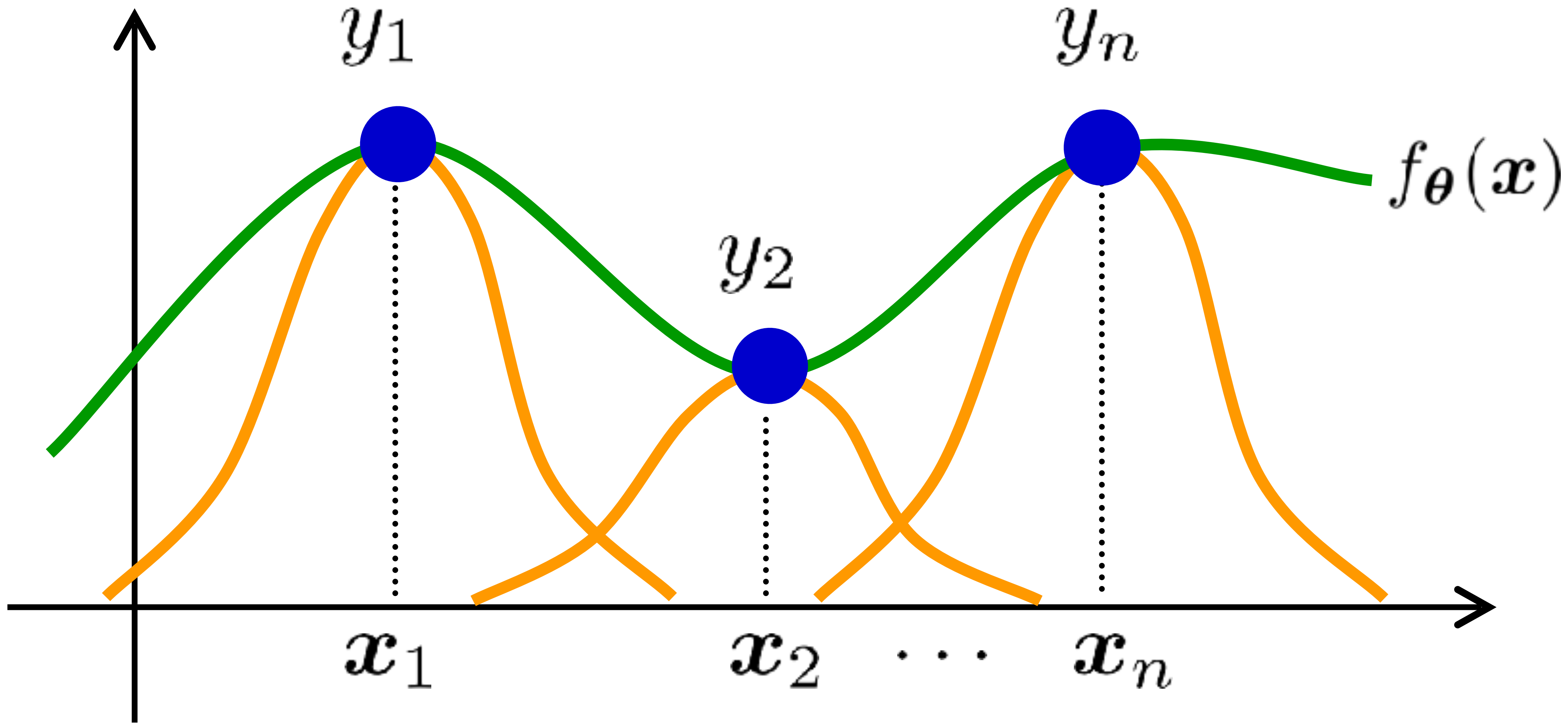
$\{\phi_j(\boldsymbol{x})\}_{j=1}^{b}$: basis functions

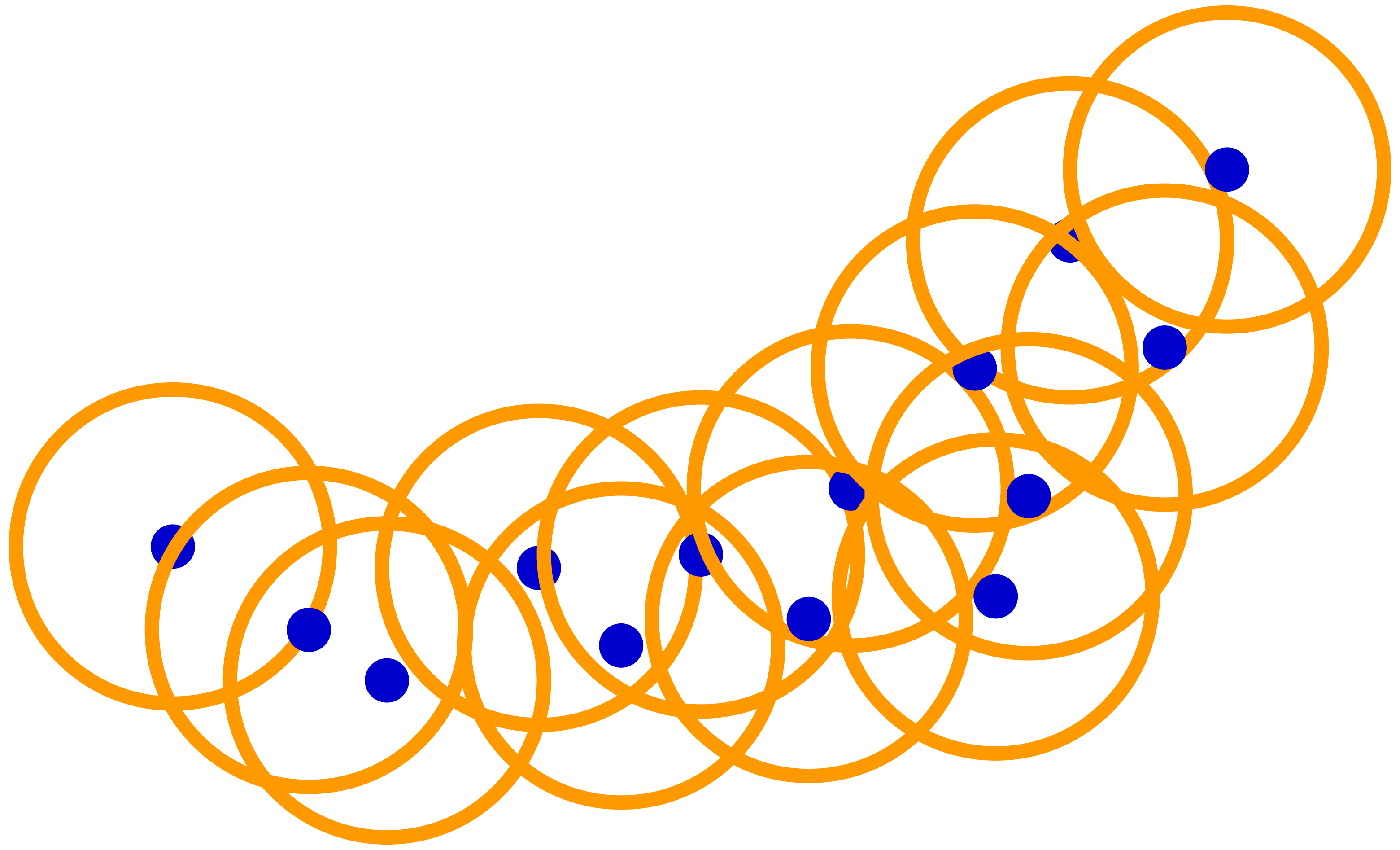
- Linear model:

Gaussian kernel

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j K(\boldsymbol{x}, \boldsymbol{x}_j)$$

$$K(\boldsymbol{x}, \boldsymbol{c}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}\|^2}{2h^2}\right)$$

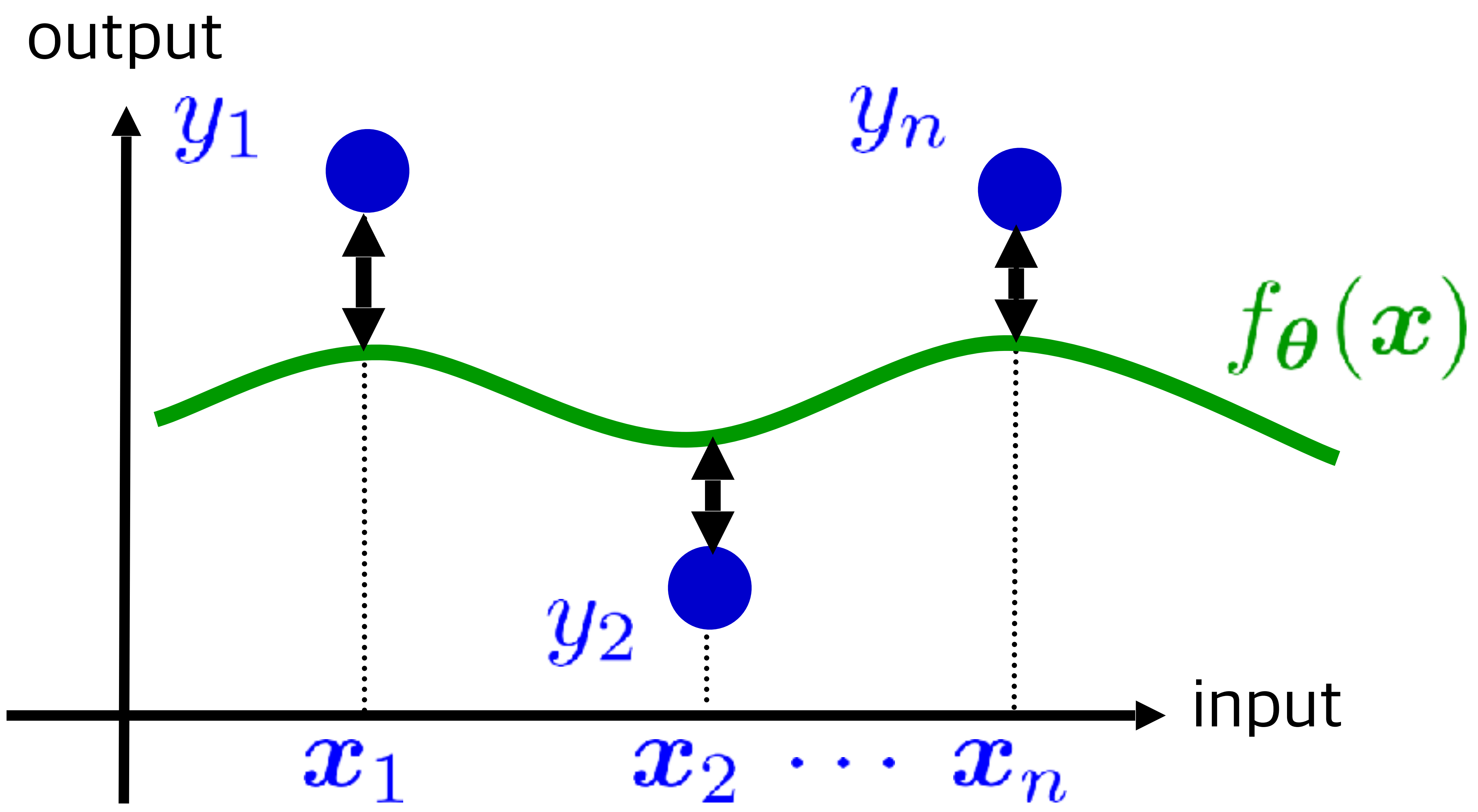




Least squares regression

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

- Minimize the squared error between training outputs:



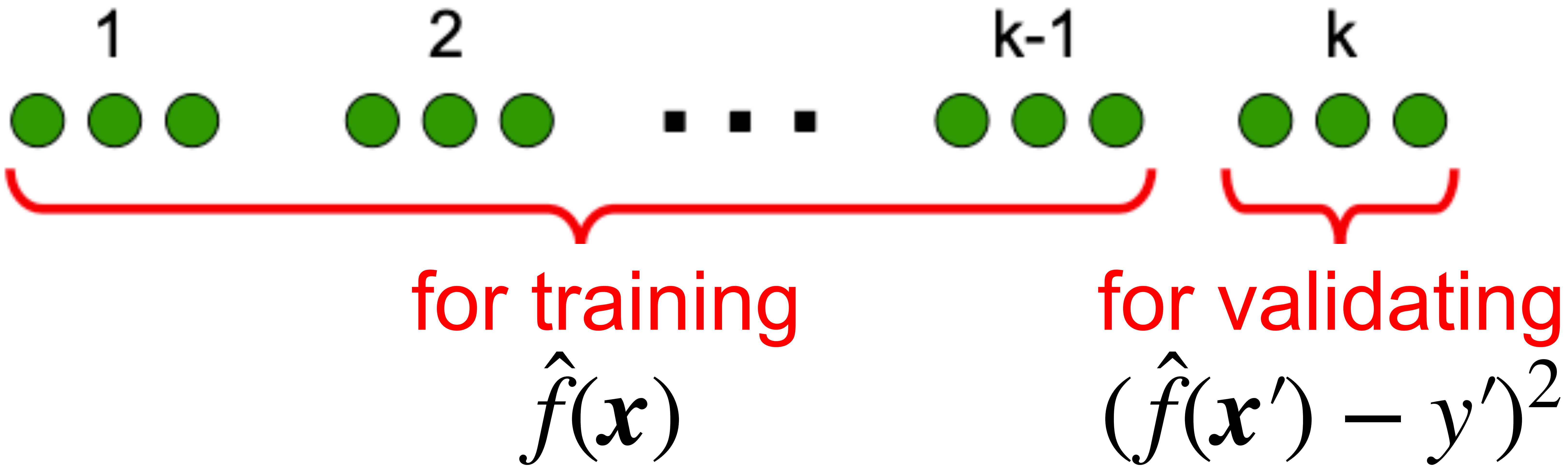
Regularization

$$\min_{\boldsymbol{\theta}} \left[\underbrace{\frac{1}{2}\sum_{i=1}^{n}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\right)^2}_{\text{goodness of fit for training output}} + \underbrace{\frac{\lambda}{2}\|\boldsymbol{\theta}\|^2}_{\text{penalty term for preventing param values from becoming too large (regularization)}} \right]$$

- The model strikes a good balance between fitting the training output well and keeping the parameter values small.
- Also called ℓ_2-Regularized regression

Cross validation

- Split training samples $Z = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ into k groups: $\{Z_i\}_{i=1}^{k}$
- Use samples from groups excluding Z_i and learn θ (fix λ, h).
- Use the remaining Z_i to check the test error.
- Repeat this for all $i \in [k]$, and return the mean of the test errors.



Schedule

1. 04/8 Introduction
2. 04/15 Regression 1
3. 04/22 Regression 2
- 04/30 Cancelled
4. 05/13 Classification 1
5. 05/20 Classification 2
6. 05/27 Deep learning 1
- 06/03 No lecture
7. 06/10 Deep learning 2
8. 06/17 Deep learning 3
9. 06/24 Semi-supervised learning
10. 07/01 Language models
11. 07/08 Representation learning 1
12. 07/15 Representation learning 2
13. 07/22 Advanced topics

Sparsity of the model

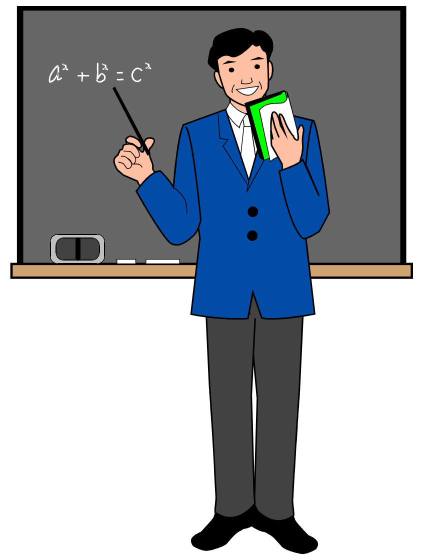
- Having a large number of parameters makes computation more challenging.
  - Example: with kernel models, it is computationally difficult when the number of training samples is large since #params = #training samples:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j K(\boldsymbol{x}, \boldsymbol{x}_j)$$

  - If many parameter values are set to zero, the computation becomes easier and more interpretable.

Sparsity of the model

- Naive approach 1:
  - Decide to not use some of the parameters.
  - When we have d params, we have 2^d different ways of choosing params.
  - Not a realistic approach when d is too large.
- Naive approach 2:
  - First use ℓ_2-regularized least squares.
  - Simply choose the params that have a small absolute value, and squash it to zero.
  - May suffer from rounding errors.



Contents

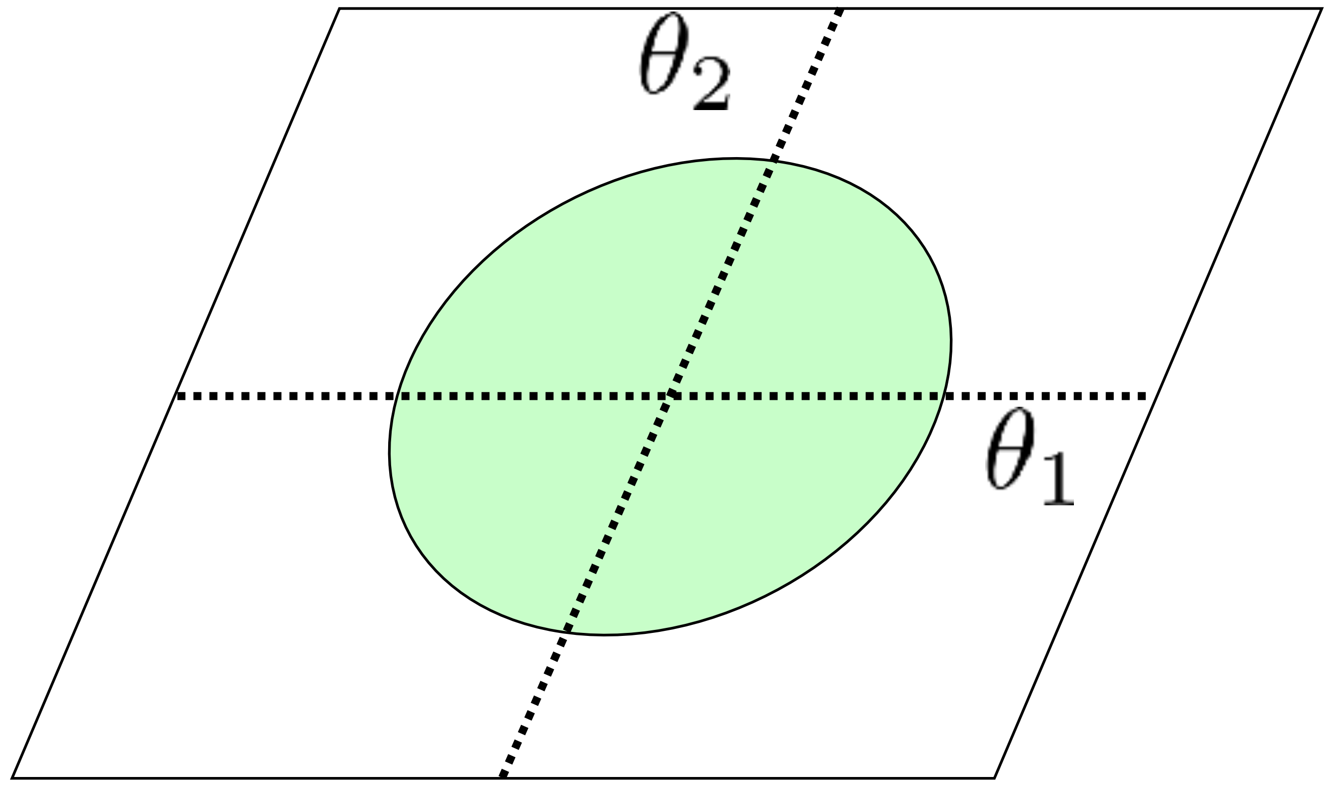
1. Sparse Regression
   1. ℓ_1-constrained least squares regression
   2. Solving ℓ_1-constrained LS
   3. Various extensions
2. Robust Regression

ℓ_1-constrained least squares regression

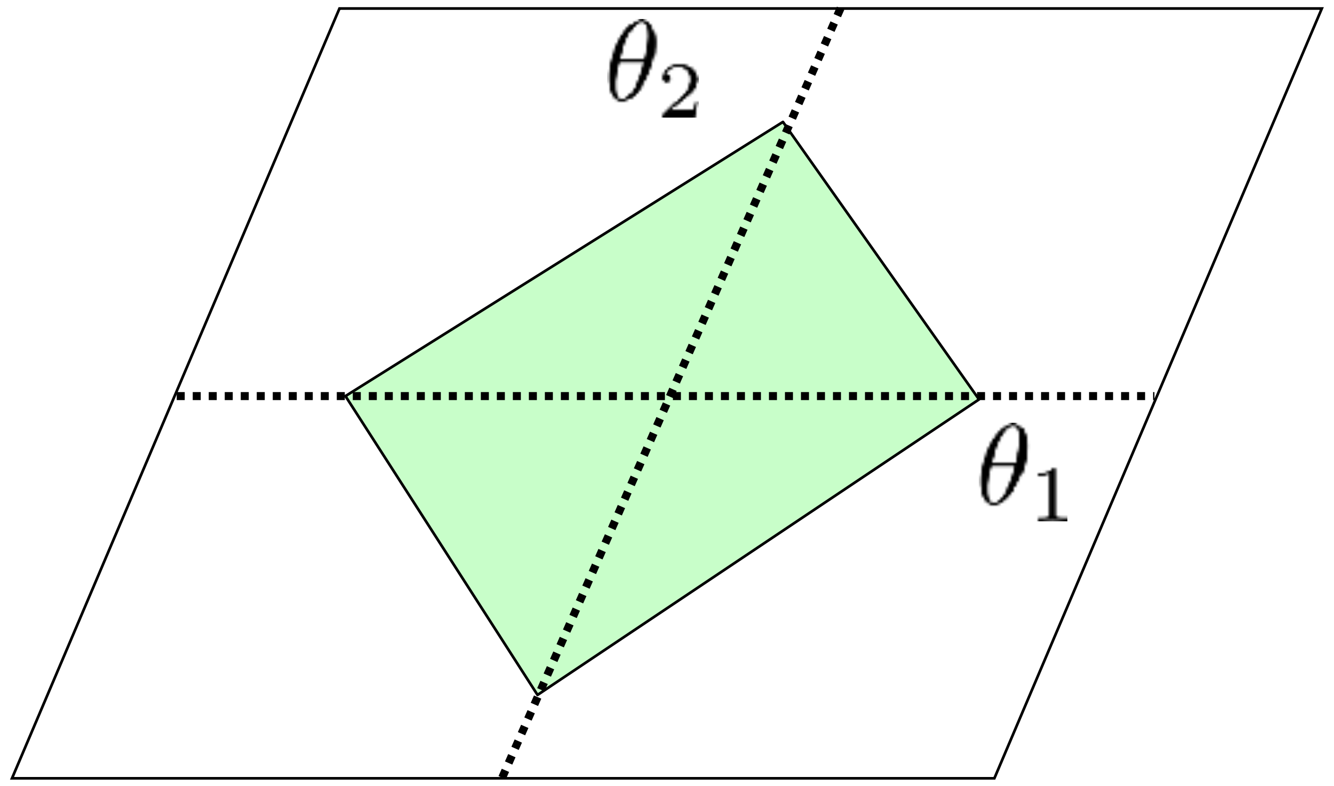
- Constrain the model to be within ℓ_1 hyper-cube

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \right)^2 \quad \text{subject to } \|\boldsymbol{\theta}\|_1 \leq R$$

$$R \geq 0 \qquad \|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{b} |\theta_j|$$



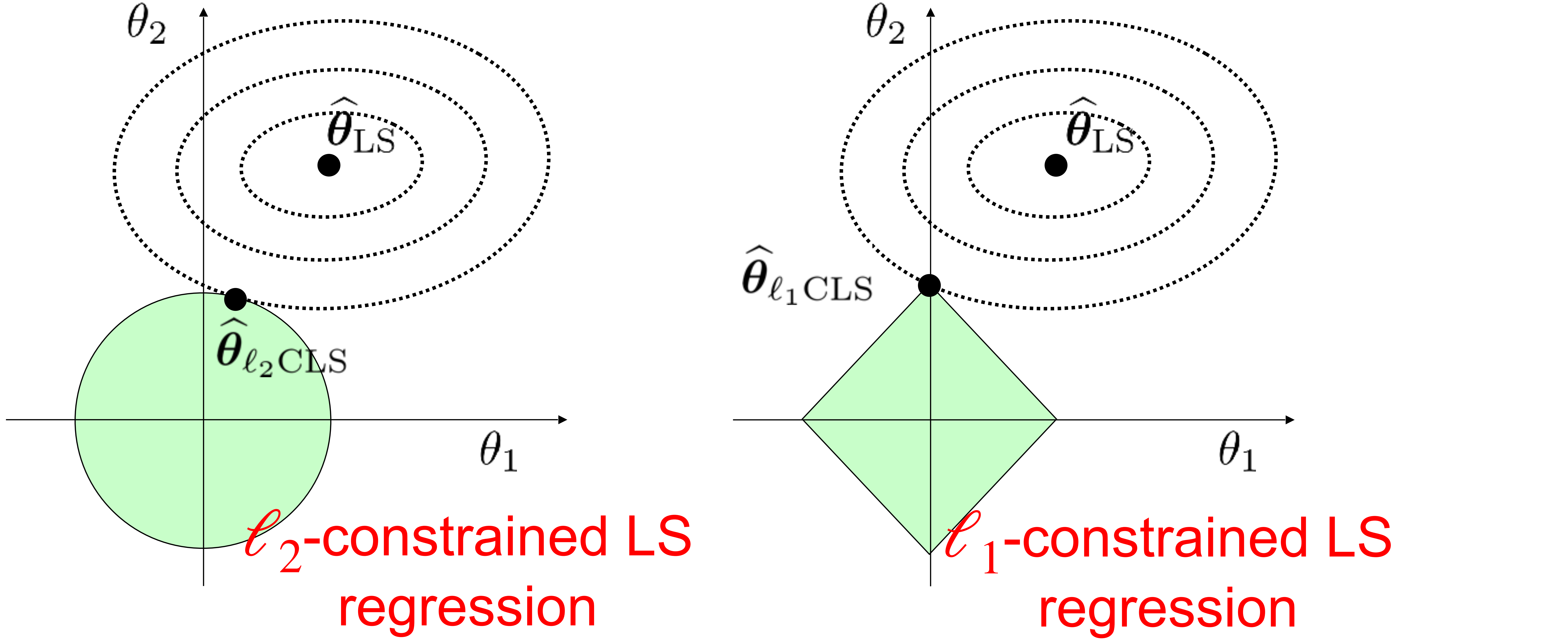
ℓ_2-constrained
LS regression



ℓ_1-constrained
LS regression

Why do we get a sparse solution?

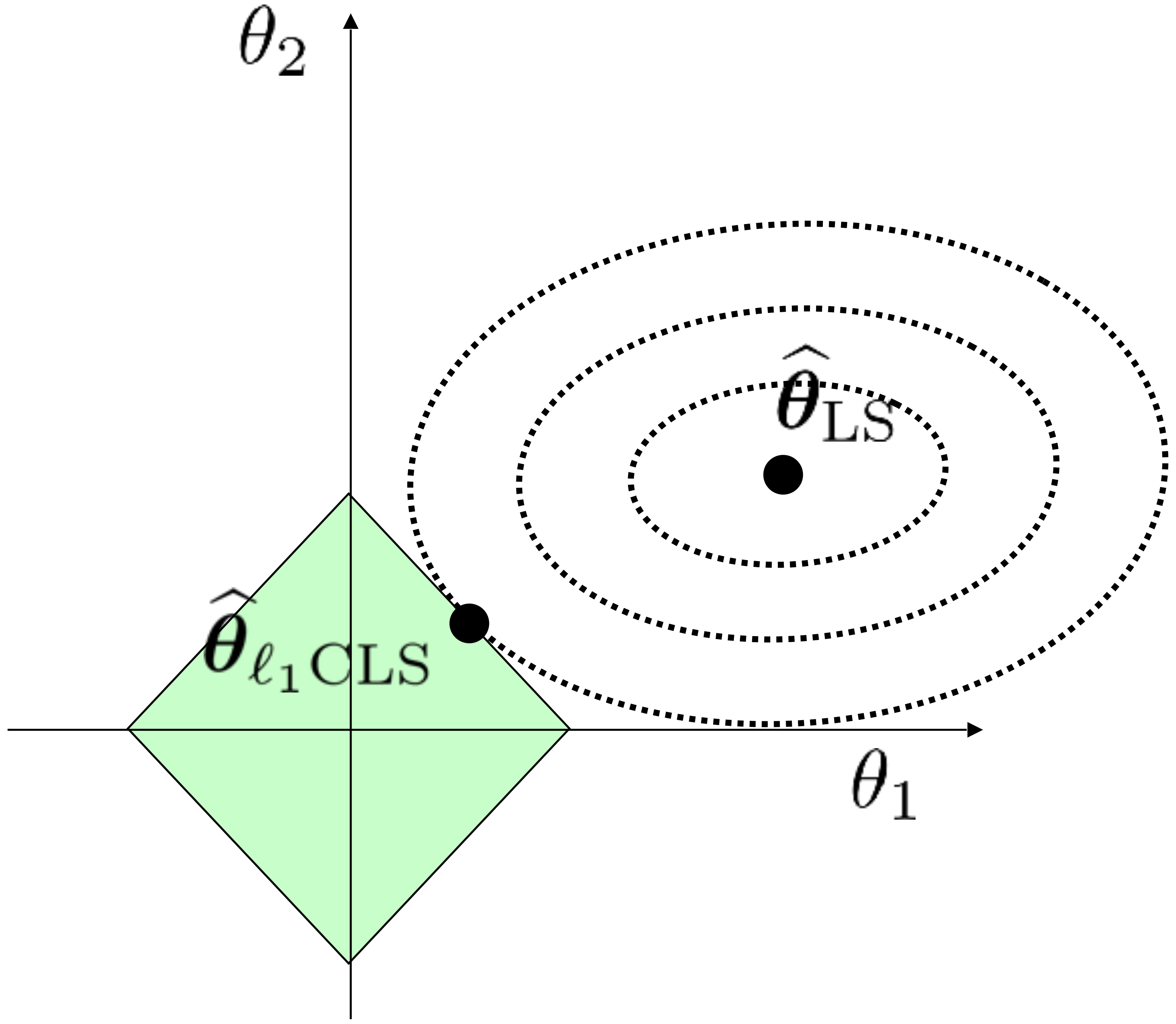
- The solution tends to be on one of the coordinate axes.



- Also called sparse regression or LASSO (Least Absolute Shrinkage and Selection Operator)

Why do we get a sparse solution?

- This does not always happen! For example, when both variables are essentially important:



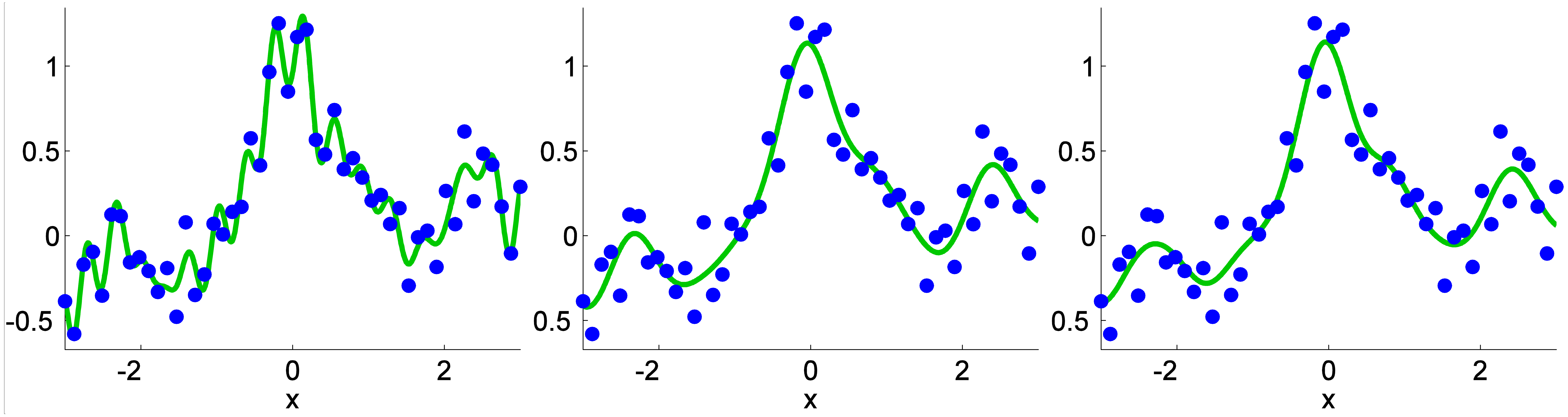
Example

- Gaussian kernel model: $$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j K(\boldsymbol{x}, \boldsymbol{x}_j)$$

$$K(\boldsymbol{x}, \boldsymbol{c}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}\|^2}{2h^2}\right)$$

Will explain algorithms soon!

Ordinary LS | ℓ_2-CLS | ℓ_1-CLS



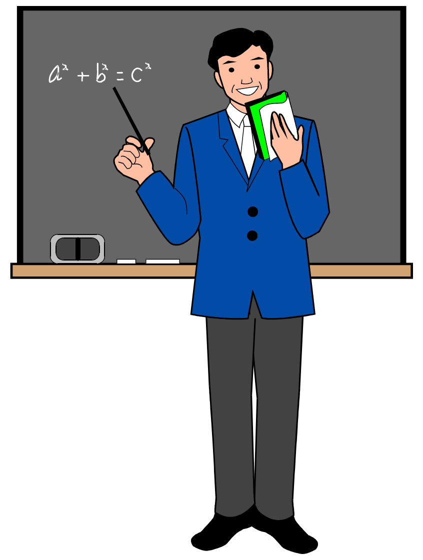
- The results of ℓ_1-CLS and ℓ_2-CLS are similar
- Out of the 50 params in ℓ_1-CLS, 38 params are zero!

Feature Selection

- If we perform sparse learning for a linear model w.r.t. the input, then some input variables will be cancelled out.

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x} \qquad \boldsymbol{x} = (x^{(1)}, \ldots, x^{(d)})^\top$$

- Will be able to select the features useful for prediction automatically.
- Example: automatic selection of important genes
- Instead of considering all 2^d combinations, we just need to decide λ in sparse learning.



Contents

1. Sparse regression
   1. ℓ_1-constrained least squares regression
   2. Solving ℓ_1-constrained LS
   3. Various extensions
2. Robust regression

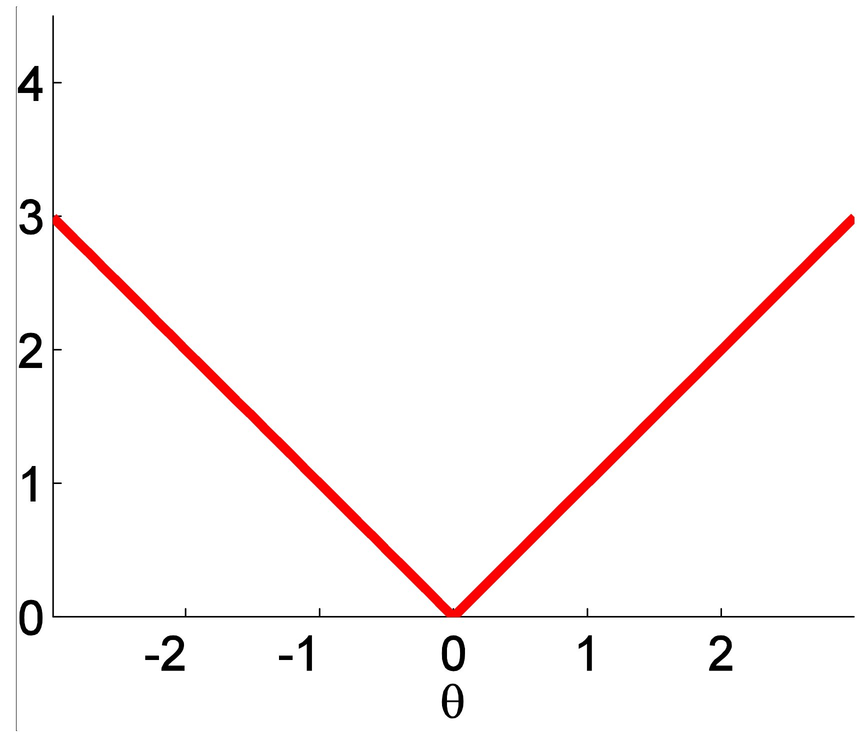
Deriving the solution

- Equivalent expression with ℓ_1 hyper-cube constraint:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[\frac{1}{2} \sum_{i=1}^{n} \Big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \Big)^2 + \lambda \sum_{j=1}^{b} |\theta_j| \right]$$

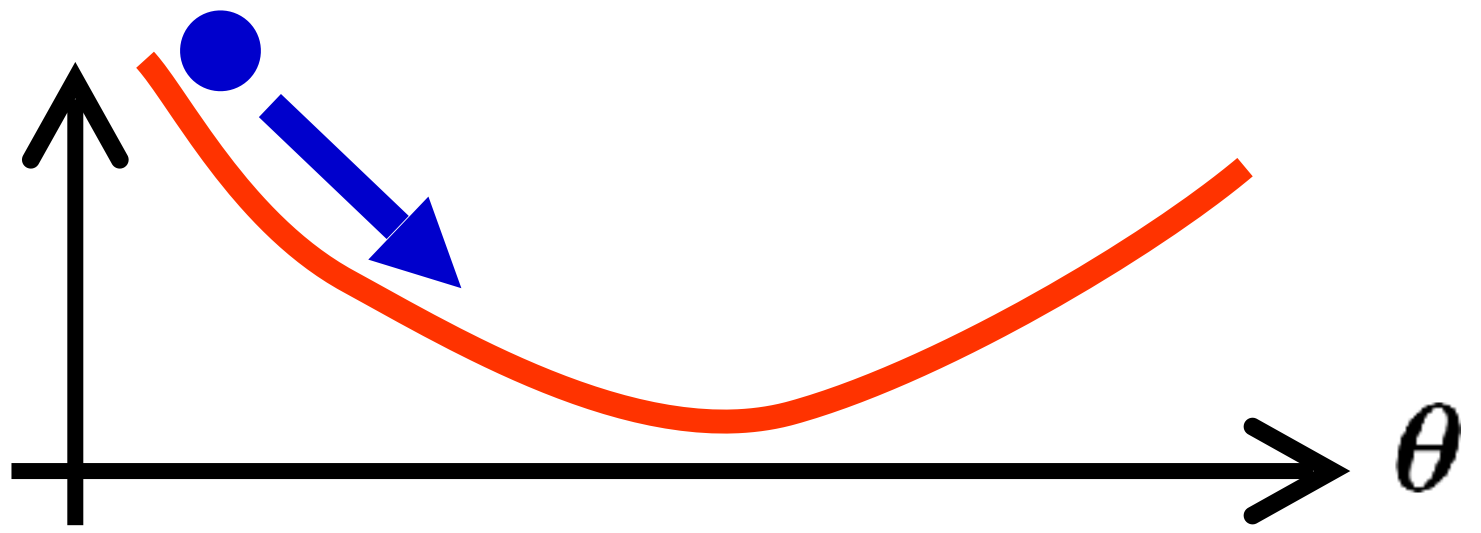
$\lambda (\geq 0)$: constant decided by R

- Instead of choosing R, we can specify λ.
- However, since the absolute value is not differentiable at the origin, the above optimization problem cannot be solved easily.

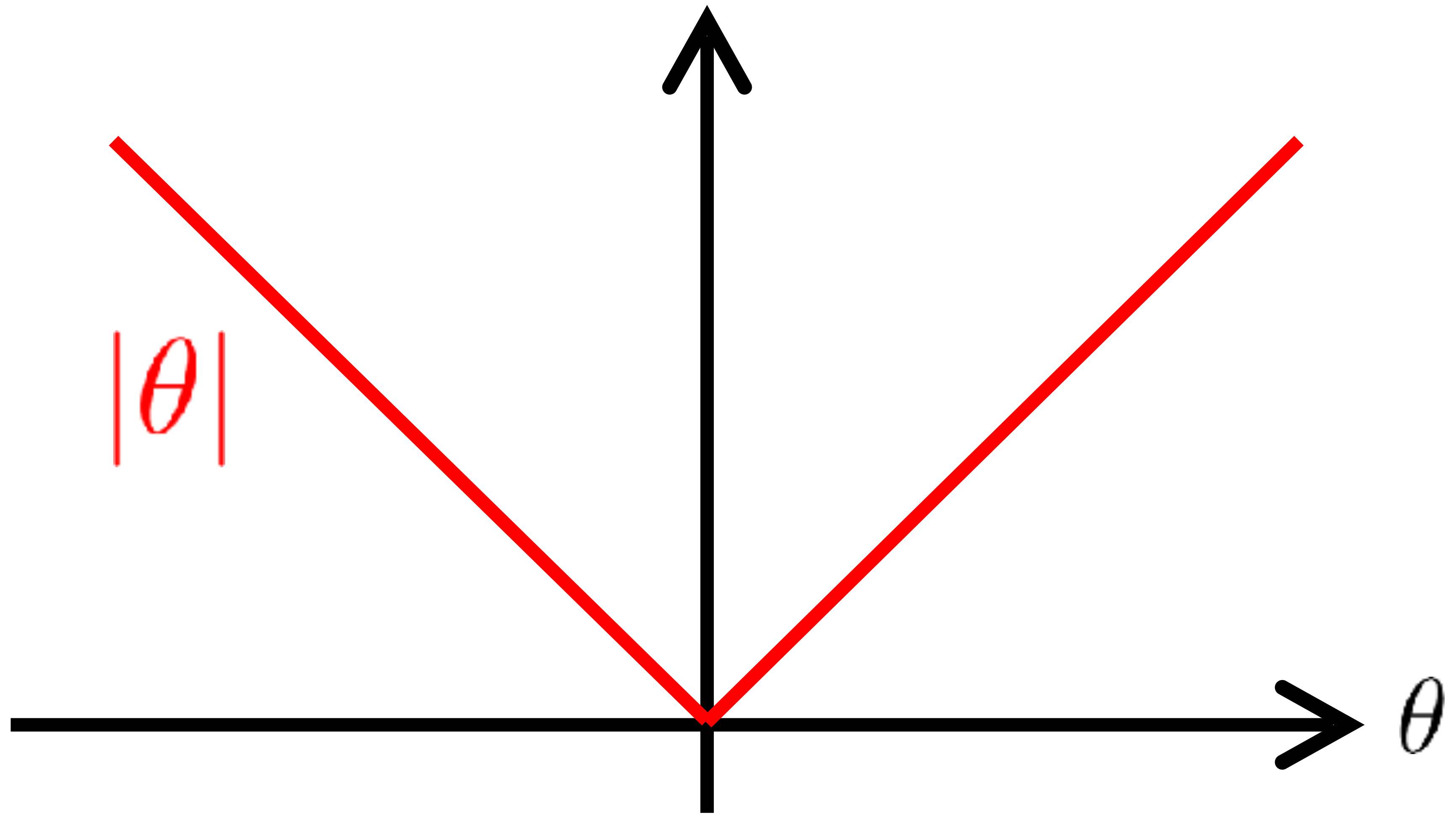


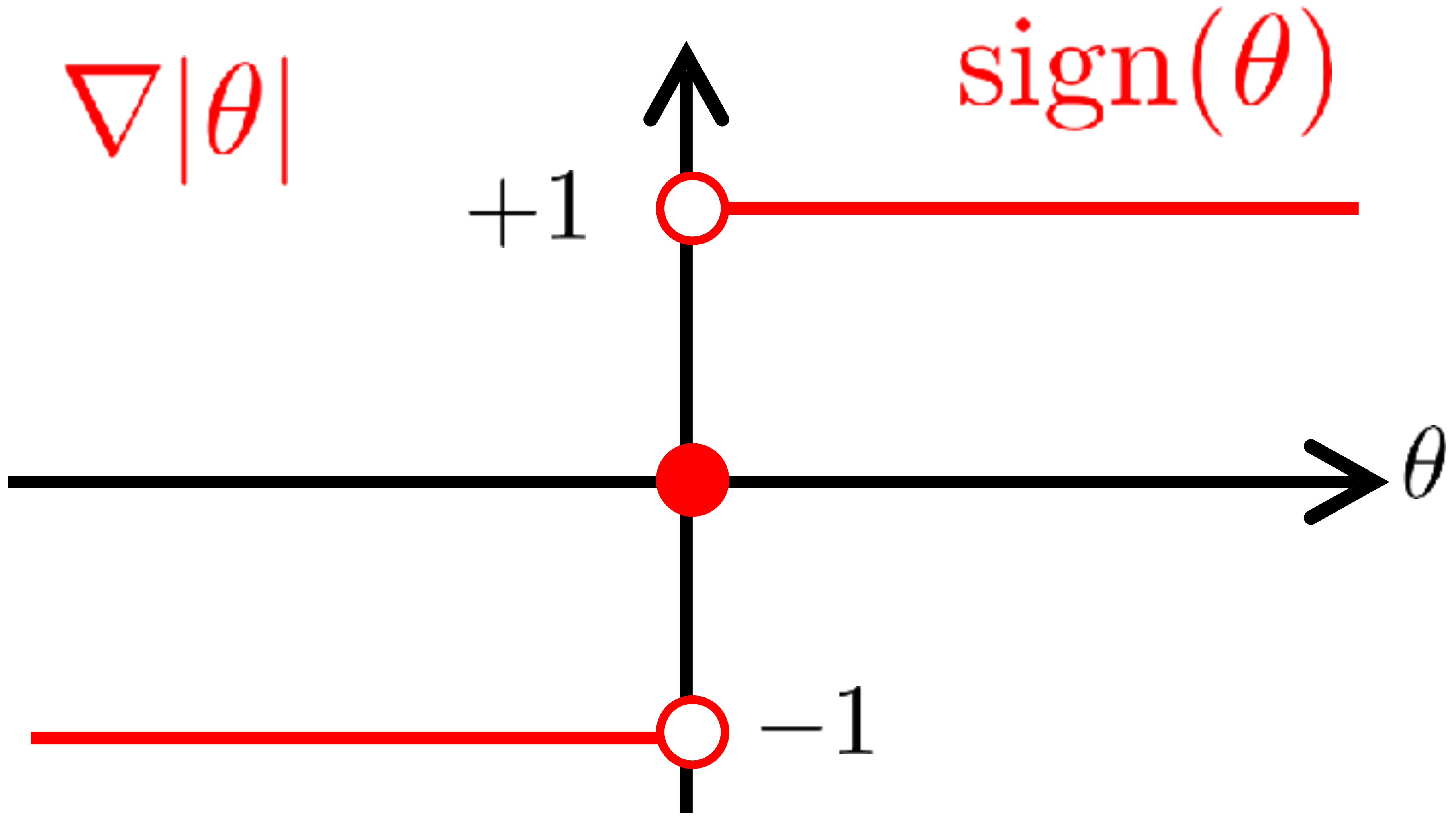
Approximate gradient method

- $\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} - \varepsilon \nabla J(\boldsymbol{\theta})$



- Approximate the derivative of the absolute value



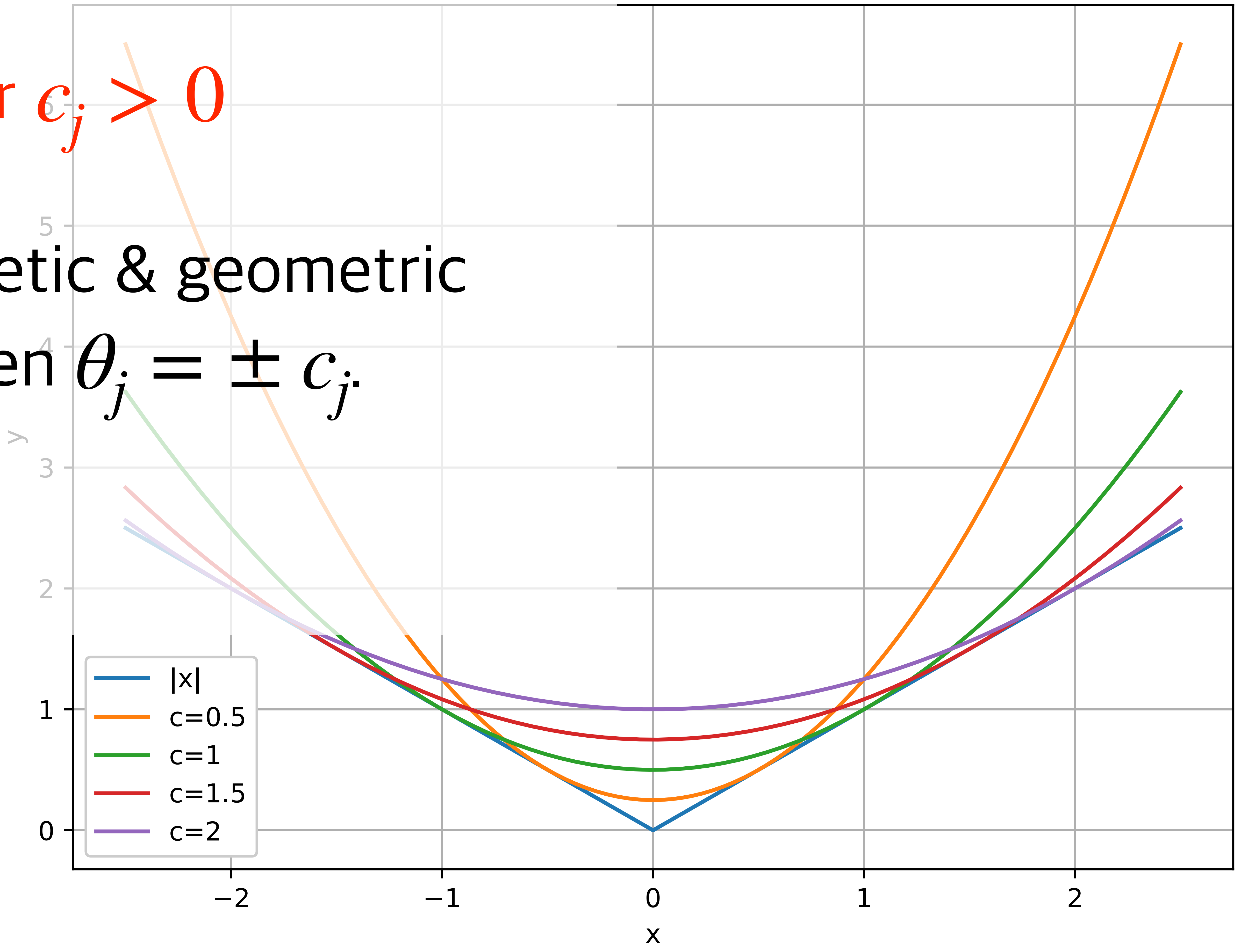


- But it is unstable and does not work well in practice because many solutions are zero.

Upper bound of ℓ_1-norm

- In order to consider a different direction, we first recall that the ℓ_1-norm is: $\|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{b} |\theta_j|$

- Upper bound: $|\theta_j| \leq \frac{\theta_j^2}{2c_j} + \frac{c_j}{2}$ for $c_j > 0$
  - Why? Apply inequality of arithmetic & geometric mean to RHS. Equality holds when $\theta_j = \pm c_j$.



Upper bound and learning objective

- If we use the current param $\tilde{\theta}_j \neq 0$ for c_j,

$$|\theta_j| \leq \frac{\theta_j^2}{2|\tilde{\theta}_j|} + \frac{|\tilde{\theta}_j|}{2}$$

- If we regard $|\theta_j| = 0$ for $\tilde{\theta}_j = 0$, then the upper bound for parameter $\tilde{\theta}_j \neq 0$:

$$|\theta_j| \leq \frac{|\tilde{\theta}_j|^{\dagger}}{2}\theta_j^2 + \frac{|\tilde{\theta}_j|}{2}$$

$|\theta|^{\dagger} = 1/|\theta|$: generalized inverse

- Original objective:

$$J(\boldsymbol{\theta}) = J_{\mathrm{LS}}(\boldsymbol{\theta}) + \lambda\|\boldsymbol{\theta}\|_1$$

- Upper bound of $J(\boldsymbol{\theta})$:

$$\tilde{J}(\boldsymbol{\theta}) = J_{\mathrm{LS}}(\boldsymbol{\theta}) + \frac{\lambda}{2}\boldsymbol{\theta}^{\top}\tilde{\boldsymbol{\Theta}}^{\dagger}\boldsymbol{\theta} + \lambda\sum_{j=1}^{b}|\tilde{\theta}_j|/2$$

$\tilde{\boldsymbol{\Theta}}$: diagonal matrix with $|\tilde{\theta}_1|, \ldots, |\tilde{\theta}_b|$ on diag $\quad$ $\tilde{\boldsymbol{\Theta}}^{\dagger}$: diagonal matrix with $|\tilde{\theta}_i|^{\dagger}$ on the diag

Minimizing $\tilde{J}(\boldsymbol{\theta})$

$$\tilde{J}(\boldsymbol{\theta}) = J_{\mathrm{LS}}(\boldsymbol{\theta}) + \frac{\lambda}{2}\boldsymbol{\theta}^{\top}\tilde{\boldsymbol{\Theta}}^{\dagger}\boldsymbol{\theta} + \text{Constant}$$

- If we are using a linear-in-parameter model $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^{\top}\phi(\boldsymbol{x})$ then:

$$J_{\mathrm{LS}}(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{\Phi}\boldsymbol{\theta} - \boldsymbol{y}\|^2.$$

  - Recall from last week: $\boldsymbol{\Phi} = [\phi_j(\boldsymbol{x}_i)]_{j,i}$ is design matrix.

- The solution of minimizing $\tilde{J}(\boldsymbol{\theta})$:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi} + \lambda\tilde{\boldsymbol{\Theta}}^{\dagger})^{-1}\boldsymbol{\Phi}^{\top}\boldsymbol{y}$$

  - Note that we can ignore $\sum_{j=1}^{b}|\tilde{\theta}_j|/2$ in the minimization problem since it is a constant w.r.t. $\boldsymbol{\theta}$.

Algorithm

- 1. Initialize parameter $\boldsymbol{\theta}$.
- 2. Form current $\boldsymbol{\theta}$, derive $\boldsymbol{\Theta}$:

$$\boldsymbol{\Theta} \longleftarrow \mathrm{diag}(|\theta_1|, \ldots, |\theta_b|).$$

- 3. Update $\boldsymbol{\theta}$:

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

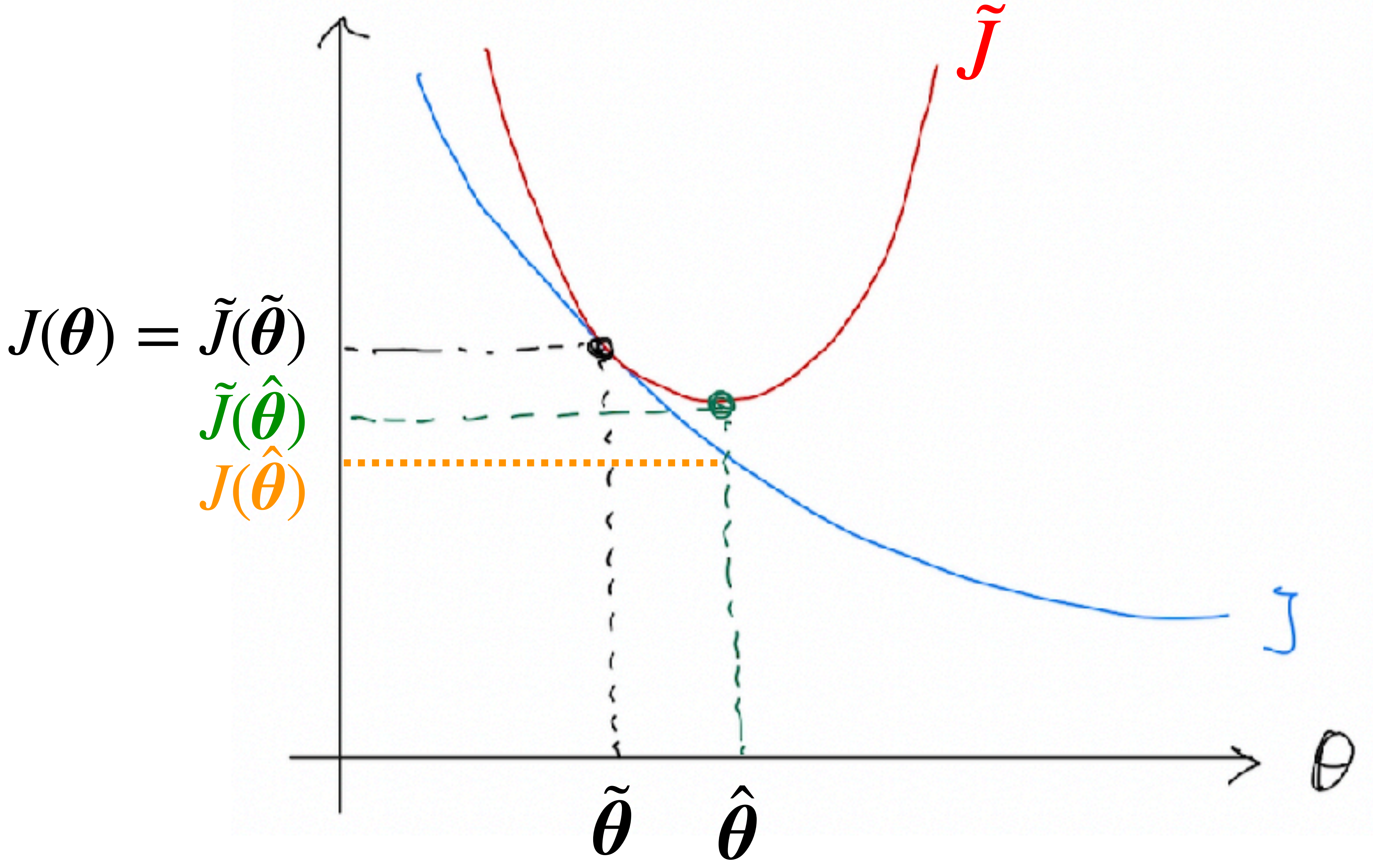
- Repeat 2 and 3 until convergence.

This is called: iteratively reweighted shrinkage

A closer look into the updates

- Some properties:
  - Since quadratic function is tangent to original function when $\boldsymbol{\theta} = \tilde{\boldsymbol{\theta}}$: $J(\tilde{\boldsymbol{\theta}}) = \tilde{J}(\tilde{\boldsymbol{\theta}})$.
  - Since $\hat{\boldsymbol{\theta}}$ is the minimizer of $\tilde{J}$: $\tilde{J}(\tilde{\boldsymbol{\theta}}) \geq \tilde{J}(\hat{\boldsymbol{\theta}})$.
  - Since $\tilde{J}(\boldsymbol{\theta})$ is upper bound of J: $\tilde{J}(\hat{\boldsymbol{\theta}}) \geq J(\hat{\boldsymbol{\theta}})$.
- To summarize:

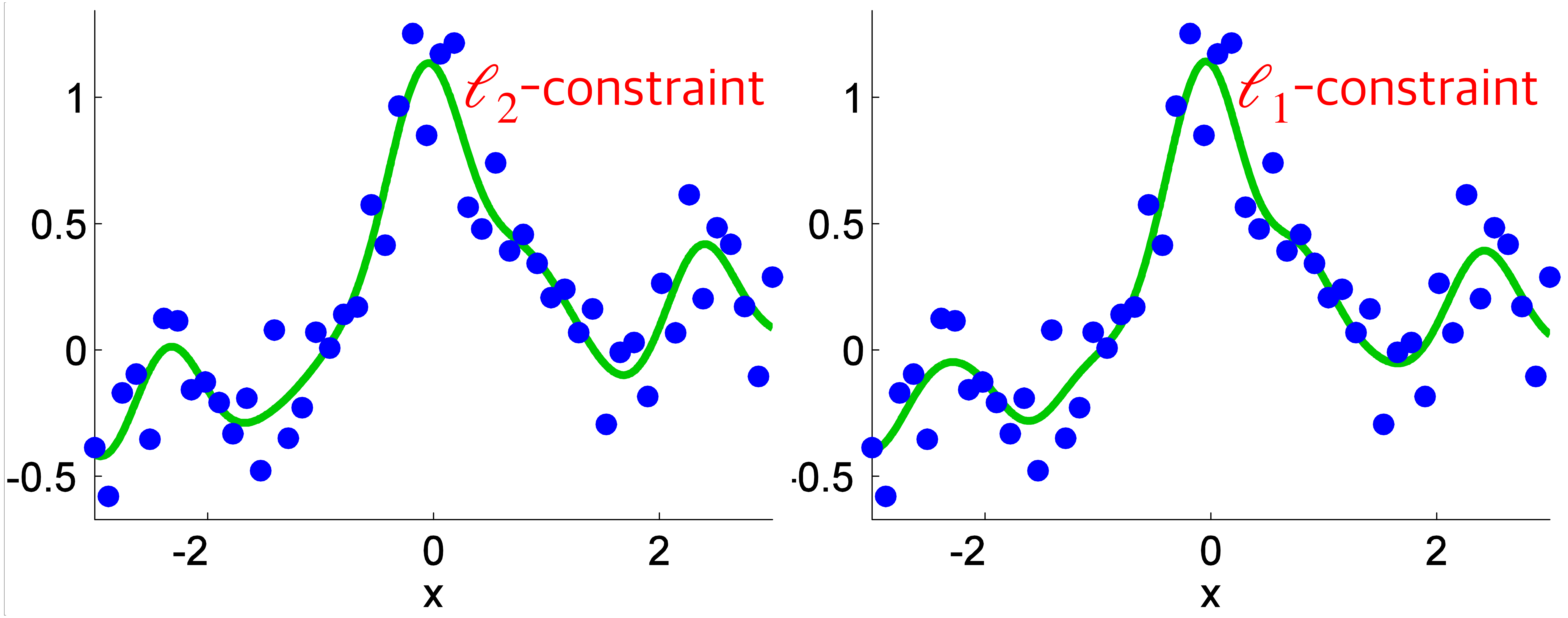
$$J(\tilde{\boldsymbol{\theta}}) = \tilde{J}(\tilde{\boldsymbol{\theta}}) \geq \tilde{J}(\hat{\boldsymbol{\theta}}) \geq J(\hat{\boldsymbol{\theta}})$$



Example

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

Implementing this is homework!

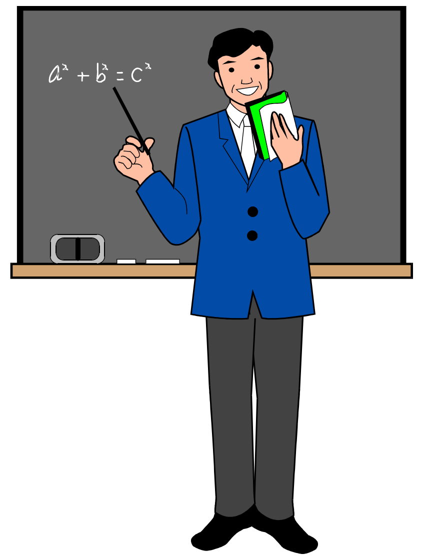


- ℓ_1-constraint results look roughly the same as ℓ_2-constraint
- However, the ℓ_1-constraint has 37 out of 50 parameters that are zero!

Note: if absolute value is below 1e-3, we regard it as zero.

Issues of iteratively reweighted shrinkage

- The cost of 1 iteration is heavy and when c_j becomes extremely small, may become unstable due to $1/c_j$.
- Still, the super simple implementation makes it a practical choice.
- Many advanced methods exists, including:
  - Accelerated proximal gradient method
  - Alternating direction method of multipliers (ADMM)



Contents

Online learning by stochastic gradient method (least squares)

$$\min_{\boldsymbol{\theta}} \frac{1}{2}\sum_{i=1}^{n}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\right)^2 \qquad f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b}\theta_j\phi_j(\boldsymbol{x})$$

1. Initialize parameter $\boldsymbol{\theta}$.
2. Gradient descent on a randomly chosen $(\boldsymbol{x}, y)$:

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} - \varepsilon\frac{\partial}{\partial\boldsymbol{\theta}}\frac{(f_{\boldsymbol{\theta}}(\boldsymbol{x}) - y)^2}{2}$$
$$= \boldsymbol{\theta} - \varepsilon\boldsymbol{\phi}(\boldsymbol{x})\left(\boldsymbol{\theta}^\top\boldsymbol{\phi}(\boldsymbol{x}) - y\right)$$

stepsize
$\varepsilon > 0$

3. Repeat step 2. until convergence

- How can we satisfy the ℓ_1 constraint?

Sparse online learning

$$\min_{\boldsymbol{\theta}} \frac{1}{2}\sum_{i=1}^{n}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)-y_i\right)^2+\lambda\sum_{j=1}^{b}|\theta_j| \qquad f_{\boldsymbol{\theta}}(\boldsymbol{x})=\sum_{j=1}^{b}\theta_j\phi_j(\boldsymbol{x})$$

1. Initialize parameter $\boldsymbol{\theta}$.
2. Gradient descent on a randomly chosen $(\boldsymbol{x}, y)$:

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} - \varepsilon\boldsymbol{\phi}(\boldsymbol{x})\left(\boldsymbol{\theta}^\top\boldsymbol{\phi}(\boldsymbol{x}) - y\right)$$

stepsize
$\varepsilon > 0$

3. Make the solution sparse:

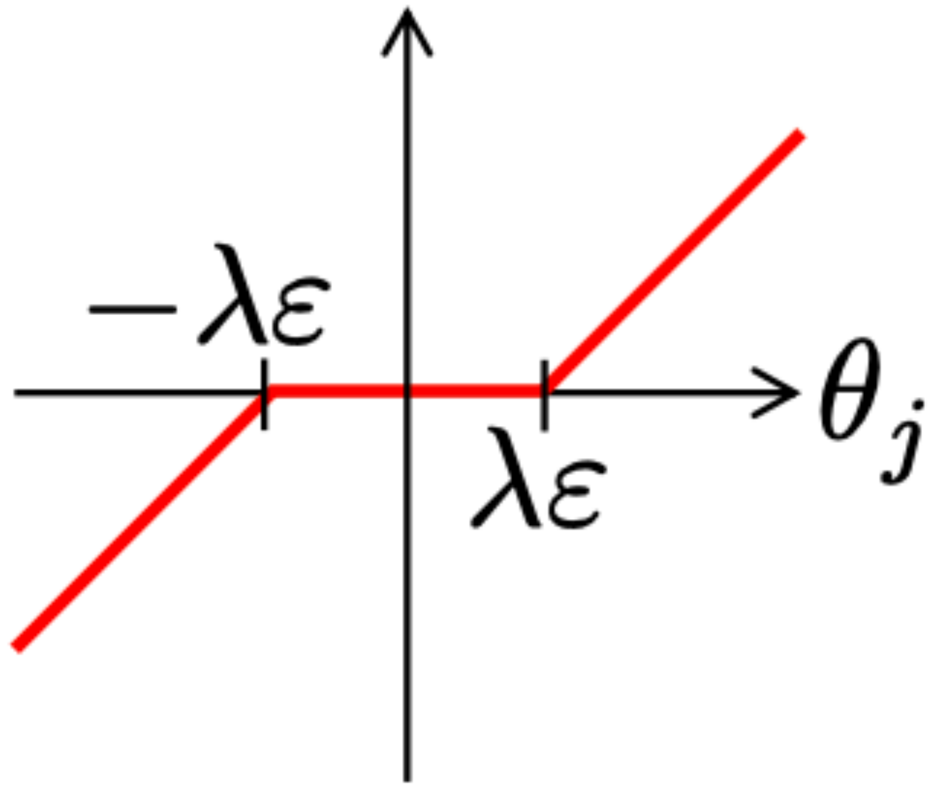
$$\forall j = 1,\ldots,b, \qquad \theta_j \longleftarrow \begin{cases} \max(0, \theta_j - \lambda\varepsilon) & (\theta_j > 0), \\ \min(0, \theta_j + \lambda\varepsilon) & (\theta_j \leq 0). \end{cases}$$

4. Repeat steps 2, 3 until convergence.

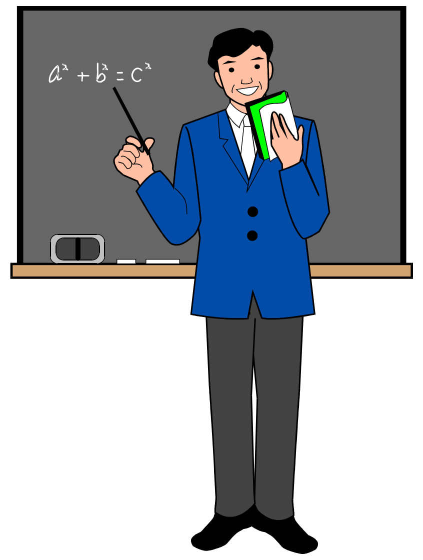
Intuition of making it sparse

$$\theta_j \longleftarrow \begin{cases} \max(0, \theta_j - \lambda\varepsilon) & (\theta_j > 0), \\ \min(0, \theta_j + \lambda\varepsilon) & (\theta_j \leq 0). \end{cases}$$

$$\max(0, \theta_j - \lambda\varepsilon) + \min(0, \theta_j + \lambda\varepsilon)$$



- With regularization, we are closer to the origin.
- The method of performing corrections corresponding to the regularization term for stochastic gradient descent is called proximal gradient method.



Contents

1. Sparse regression
   1. ℓ_1 -constrained least squares regression
   2. Solving ℓ_1-constrained LS
   3. Various extensions
      - A) Online learning
      - B) Generalized ℓ_1-norm
      - C) ℓ_p-norm
      - D) $\ell_1 + \ell_2$-norm
      - E) $\ell_{1,2}$-norm
2. Robust regression

Generalized ℓ_1-norm

$$\|\boldsymbol{F\theta}\|_1 = \sum_j \left| \sum_{j'} F_{j,j'} \theta_{j'} \right|$$

- Example: the norm of the difference of adjacent elements

$$\sum_j |\theta_{j+1} - \theta_j| \qquad F_{j,j'} = \begin{cases} 1 & (j' = j+1) \\ -1 & (j' = j) \\ 0 & (\text{otherwise}) \end{cases}$$

- The generalized ℓ_1 version of LS is called fused lasso.

Generalized ℓ_1-constrained least squares regression

$$\min_{\boldsymbol{\theta}} \frac{1}{2}\sum_{i=1}^{n}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\right)^2 \quad \text{subject to } \|\boldsymbol{F}\boldsymbol{\theta}\|_1 \leq R$$

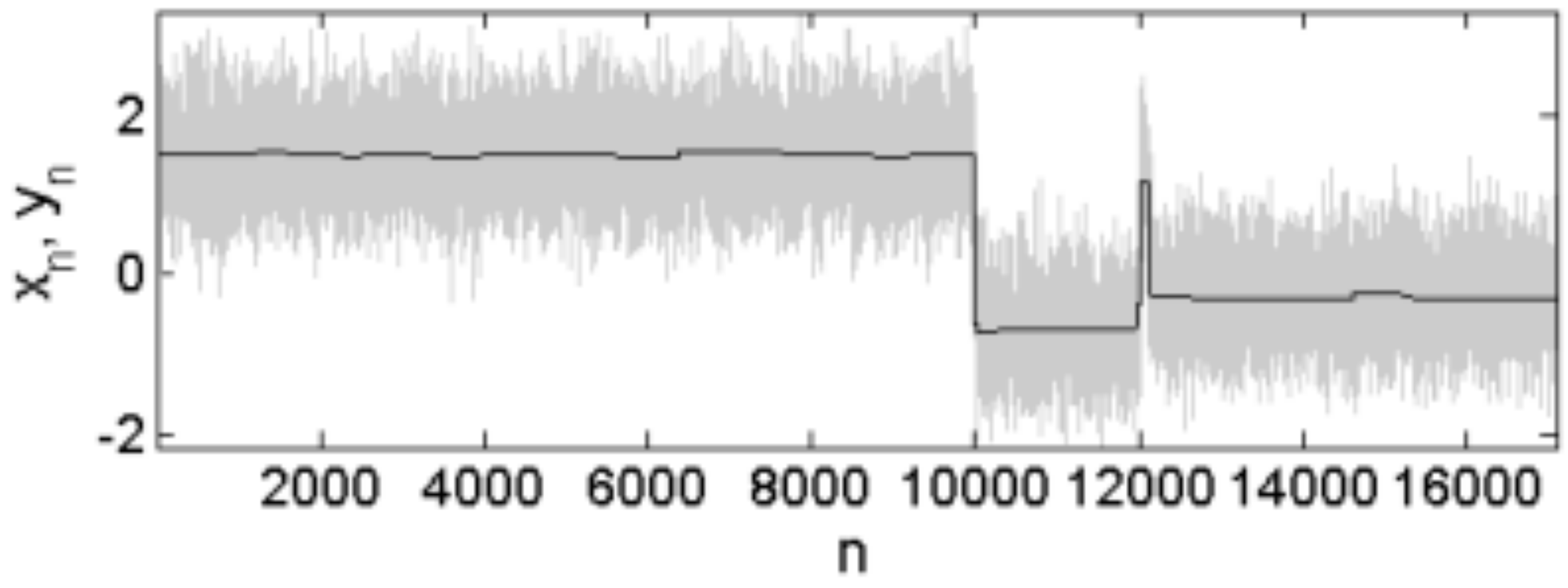
$$R \geq 0$$

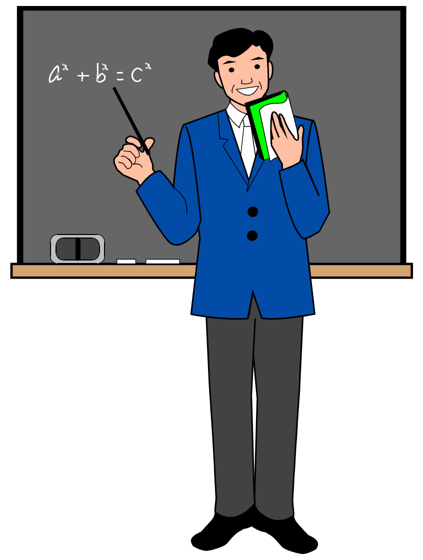
- Example: total variation noise removal

$$\min_{\boldsymbol{\theta}} \frac{1}{2}\|\boldsymbol{\theta} - \boldsymbol{y}\|^2 \quad \text{subject to } \sum_j |\theta_{j+1} - \theta_j| \leq R$$

From Wikipedia

$$F_{j,j'} = \begin{cases} 1 & (j' = j+1) \\ -1 & (j' = j) \\ 0 & (\text{otherwise}) \end{cases}$$





Contents

1. Sparse regression
   1. ℓ_1 -constrained least squares regression
   2. Solving ℓ_1-constrained LS
   3. Various extensions
      A) Online learning
      B) Generalized ℓ_1-norm
      C) ℓ_p-norm
      D) $\ell_1 + \ell_2$-norm
      E) $\ell_{1,2}$-norm
2. Robust regression

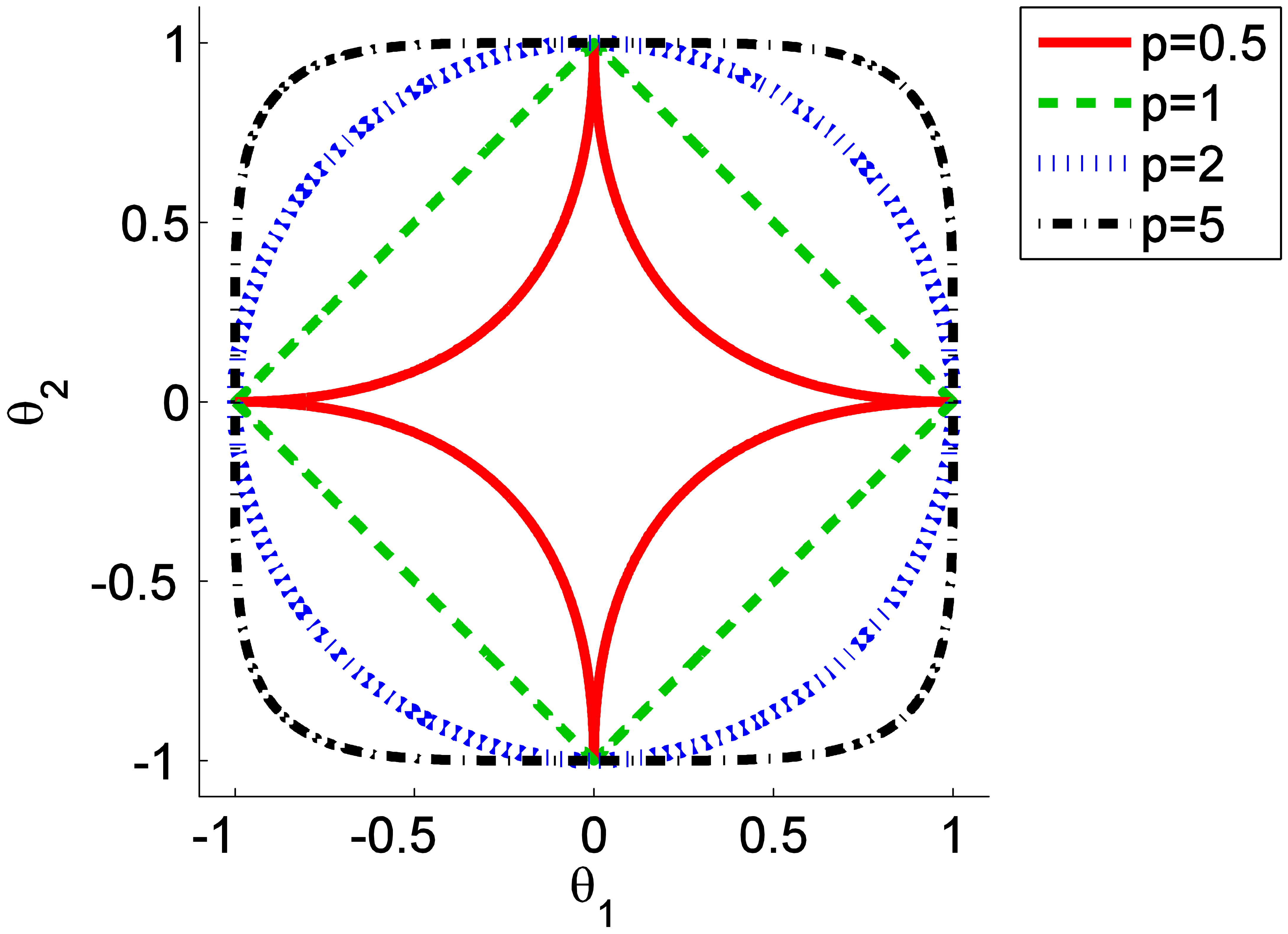
ℓ_p-norm

$$\|\boldsymbol{\theta}\|_p = \left(\sum_{j=1}^{b} |\theta_j|^p\right)^{\frac{1}{p}}$$

$p \geq 1$

$$\|\boldsymbol{\theta}\|_p = \sum_{j=1}^{b} |\theta_j|^p$$

$p \leq 1$



$p = 0$

$$\|\boldsymbol{\theta}\|_0 = \#\ \text{non-zero elements}$$

$p = \infty$

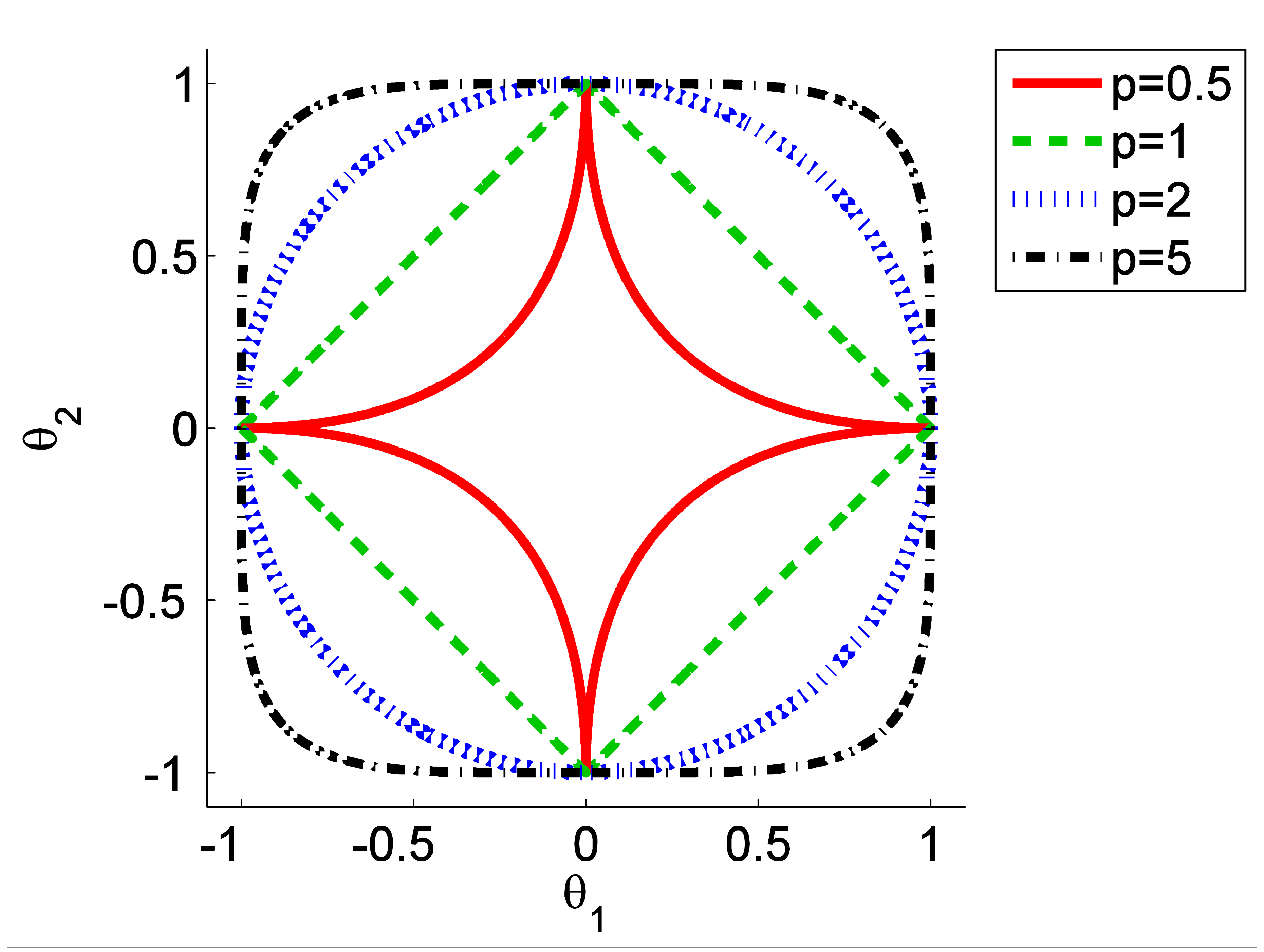
$$\|\boldsymbol{\theta}\|_\infty = \max\left\{|\theta_1|, \ldots, |\theta_b|\right\}$$

ℓ_p-constrained least squares regression

- Restrict the model to be within ℓ_p-hypercube

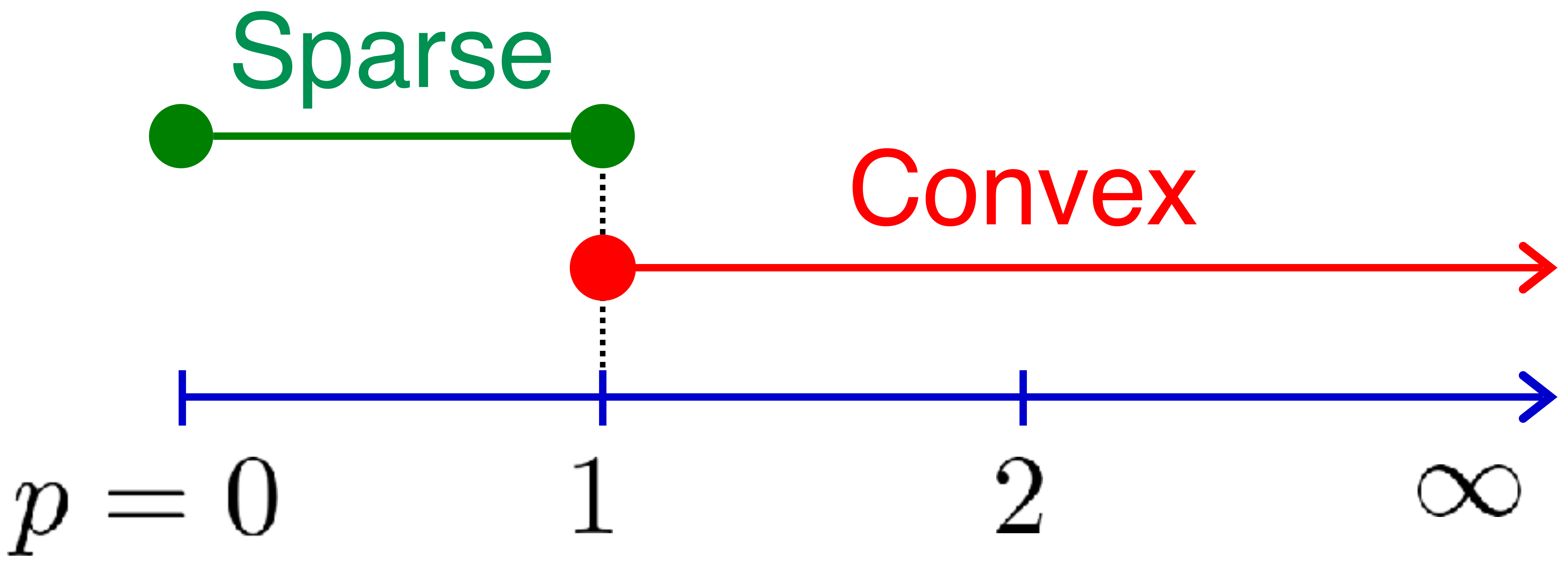
$$\min_{\boldsymbol{\theta}} \frac{1}{2}\sum_{i=1}^{n}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\right)^2 \quad \text{subject to } \|\boldsymbol{\theta}\|_p \le R$$

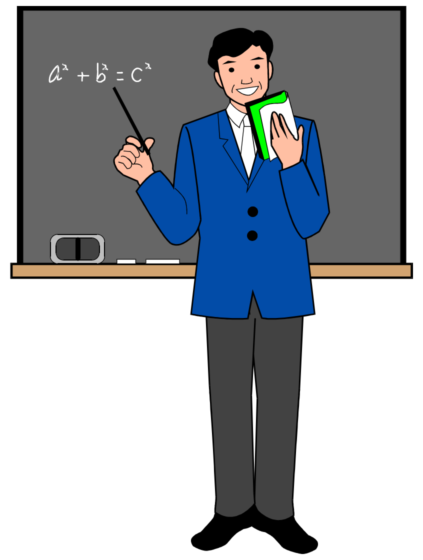
$p \geq 0$



ℓ_p-constrained least squares regression

- The solution becomes sparse: $0 \leq p \leq 1$
- Optimization problem is convex: $p \geq 1$
  (Easier to achieve global solution)
- Only $p = 1$ satisfies both!





Contents

Issue of ℓ_1-constrained LS regression

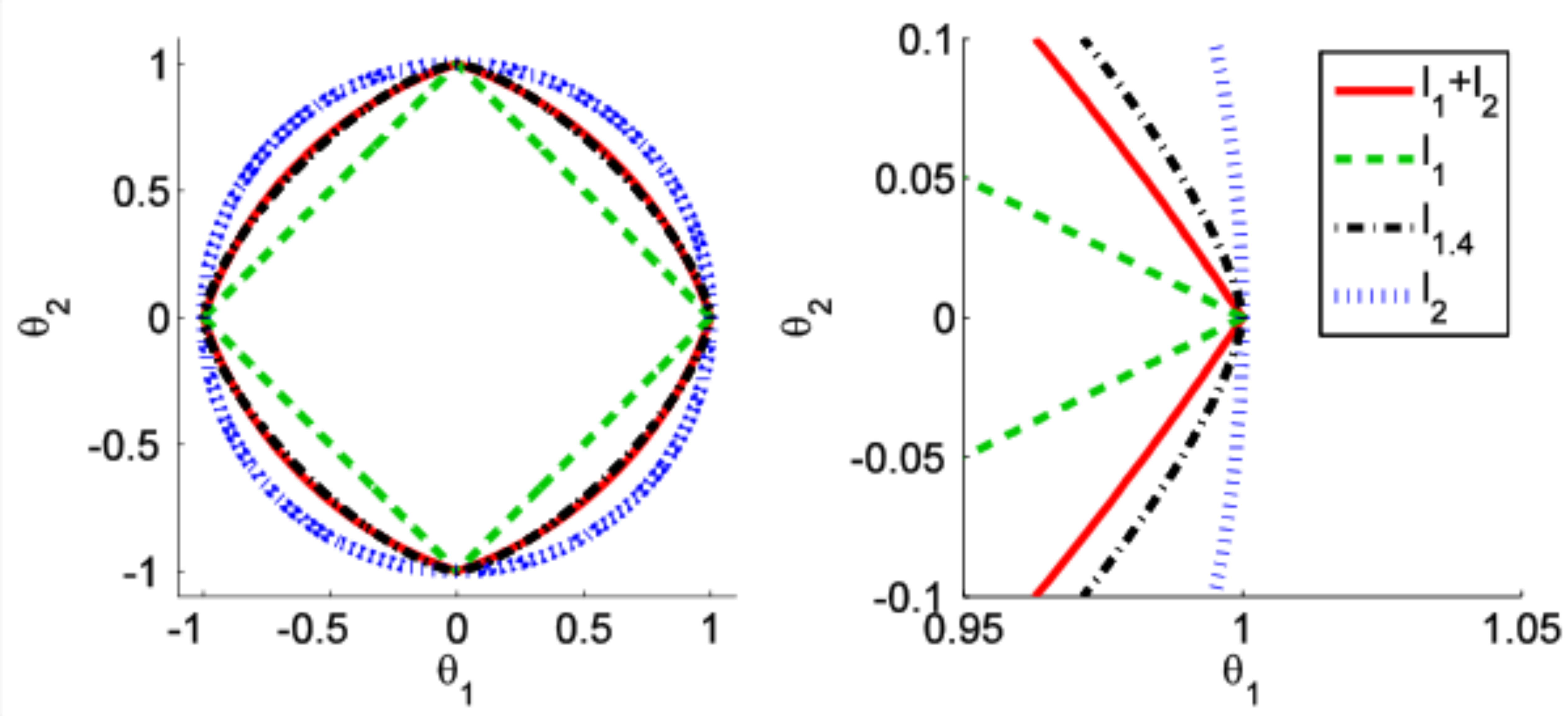
$$\min_{\boldsymbol{\theta}} \frac{1}{2}\sum_{i=1}^{n}\left(\sum_{j=1}^{b}\theta_j\phi_j(\boldsymbol{x}_i)-y_i\right)^2 \quad \text{subject to} \sum_{j=1}^{b}|\theta_j| \leq R$$

- When some basis functions $\{\phi_j(\boldsymbol{x})\}_{j=1}^{b}$ are similar, only one of them is chosen.
- When $b < n$, this may perform worse compared with ℓ_2-constrained LS regression.

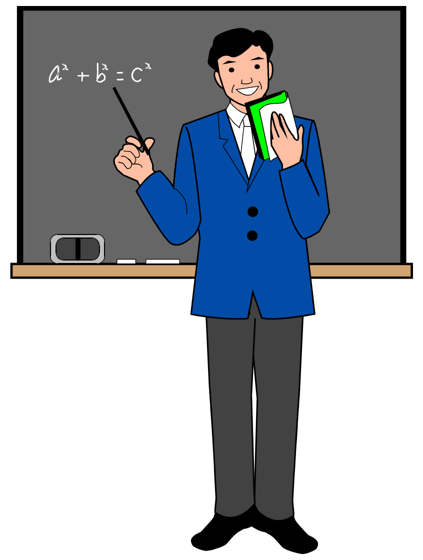
$\ell_1 + \ell_2$-constrained least squares regression

$$(1-\tau)\sum_{j=1}^{b}|\theta_j| + \tau\sum_{j=1}^{b}\theta_j^2 \le R \qquad 0 \le \tau < 1$$

- Similar to $\ell_{1.4}$-cube, but $\ell_1 + \ell_2$-cube is sharp



- Also called elastic net.



Contents

$\ell_{1,2}$-norm

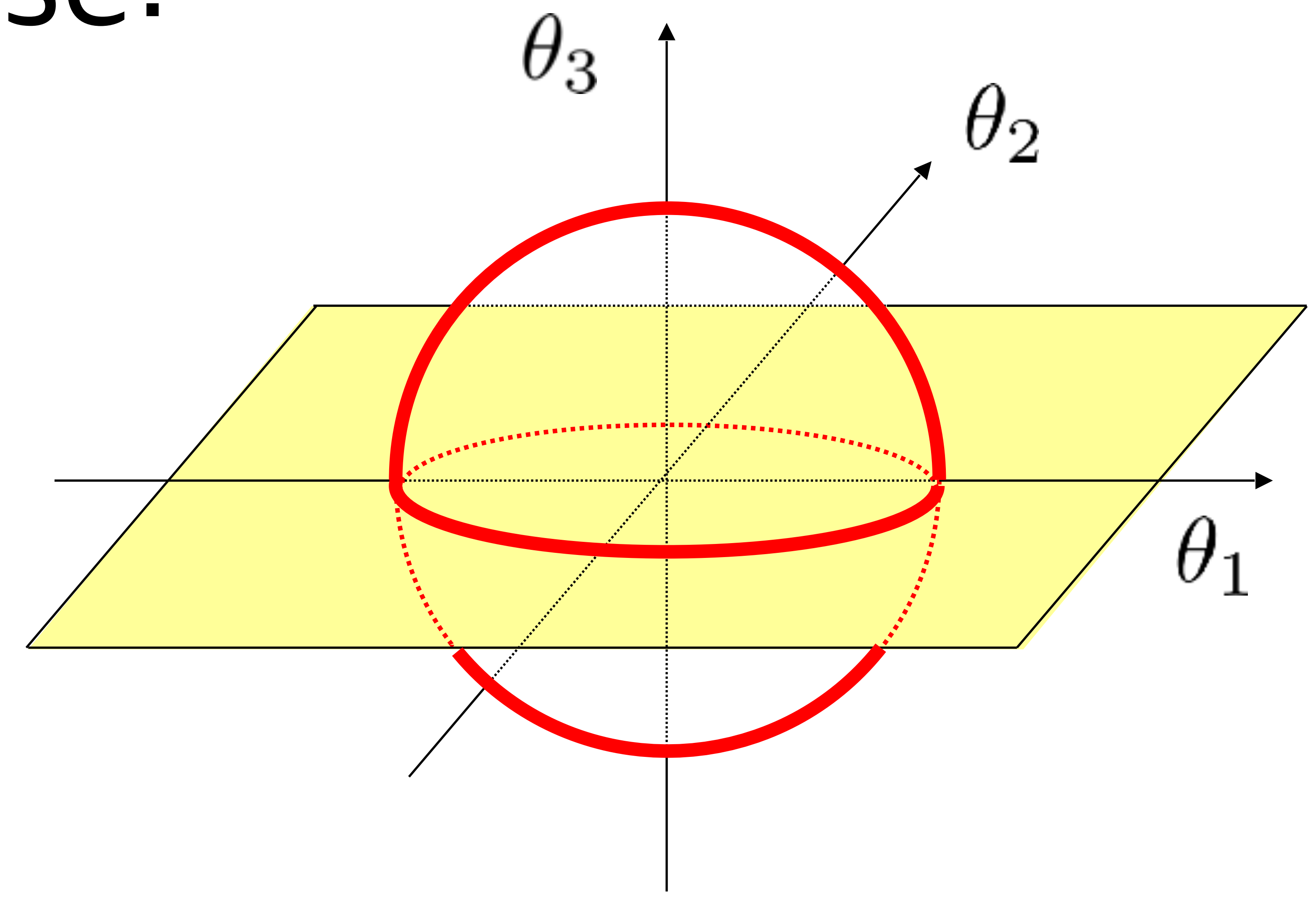
- Setup: there are variable groups that are similar, but we do not know whether they are helpful.
- When $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_b)^\top$ has a group structure $\boldsymbol{\theta} = \left(\boldsymbol{\theta}^{(1)\top}, \ldots, \boldsymbol{\theta}^{(t)\top}\right)^\top$, then the following is called $\ell_{1,2}$-norm:

$$\|\boldsymbol{\theta}\|_{1,2} = \sum_{j=1}^{t} \|\boldsymbol{\theta}^{(j)}\|_2 \qquad \boldsymbol{\theta}^{(j)} \in \mathbb{R}^{b_j}$$

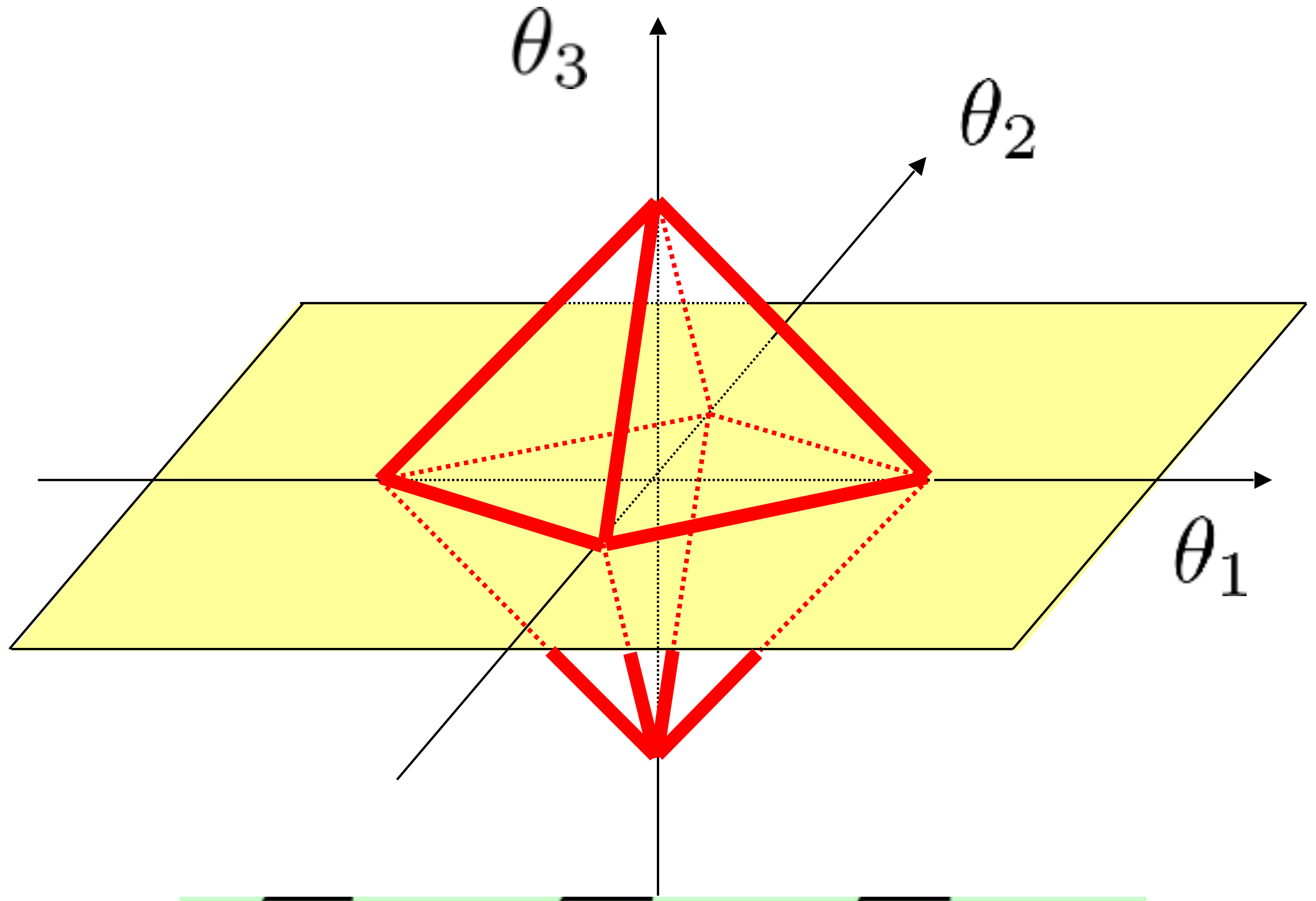
- Called group regularization when this is used for the regularization term.

Exercise

- The constraint of ℓ_2-norm and ℓ_1-norm in 3-dimensional case:



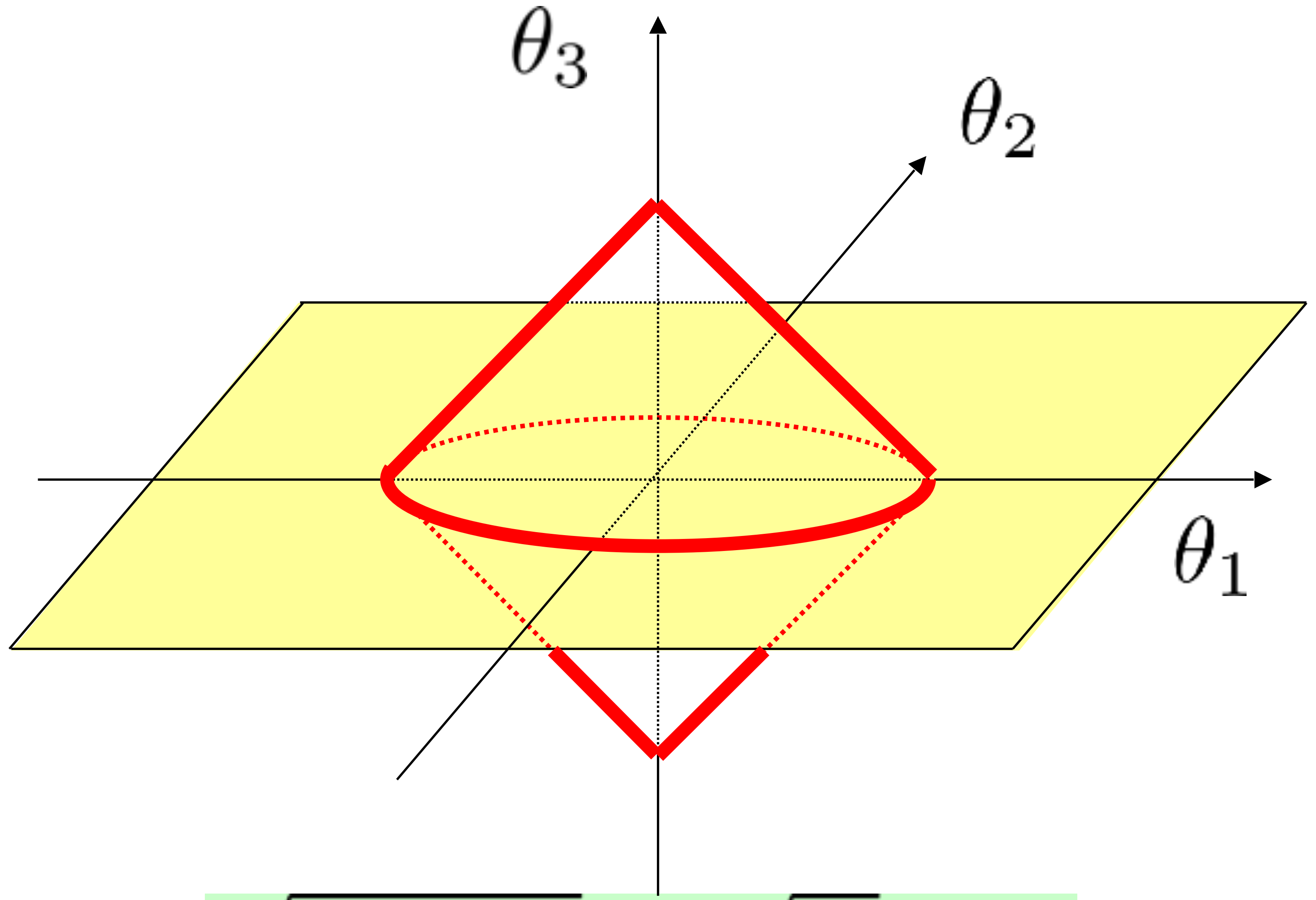
$$\sqrt{\theta_1^2 + \theta_2^2 + \theta_3^2} \leq R$$



$$\sqrt{\theta_1^2} + \sqrt{\theta_2^2} + \sqrt{\theta_3^2} \leq R$$

- Similarly, visualize the constraint of $\ell_{1,2}$-norm and the sparse solution: $\sqrt{\theta_1^2 + \theta_2^2} + \sqrt{\theta_3^2} \leq R$

Visualization



$$\sqrt{\theta_1^2 + \theta_2^2} + \sqrt{\theta_3^2} \leq R$$

- Achieve sparsity at the group level!

$$\sum_{j=1}^{t} \|\boldsymbol{\theta}^{(j)}\| \leq R$$

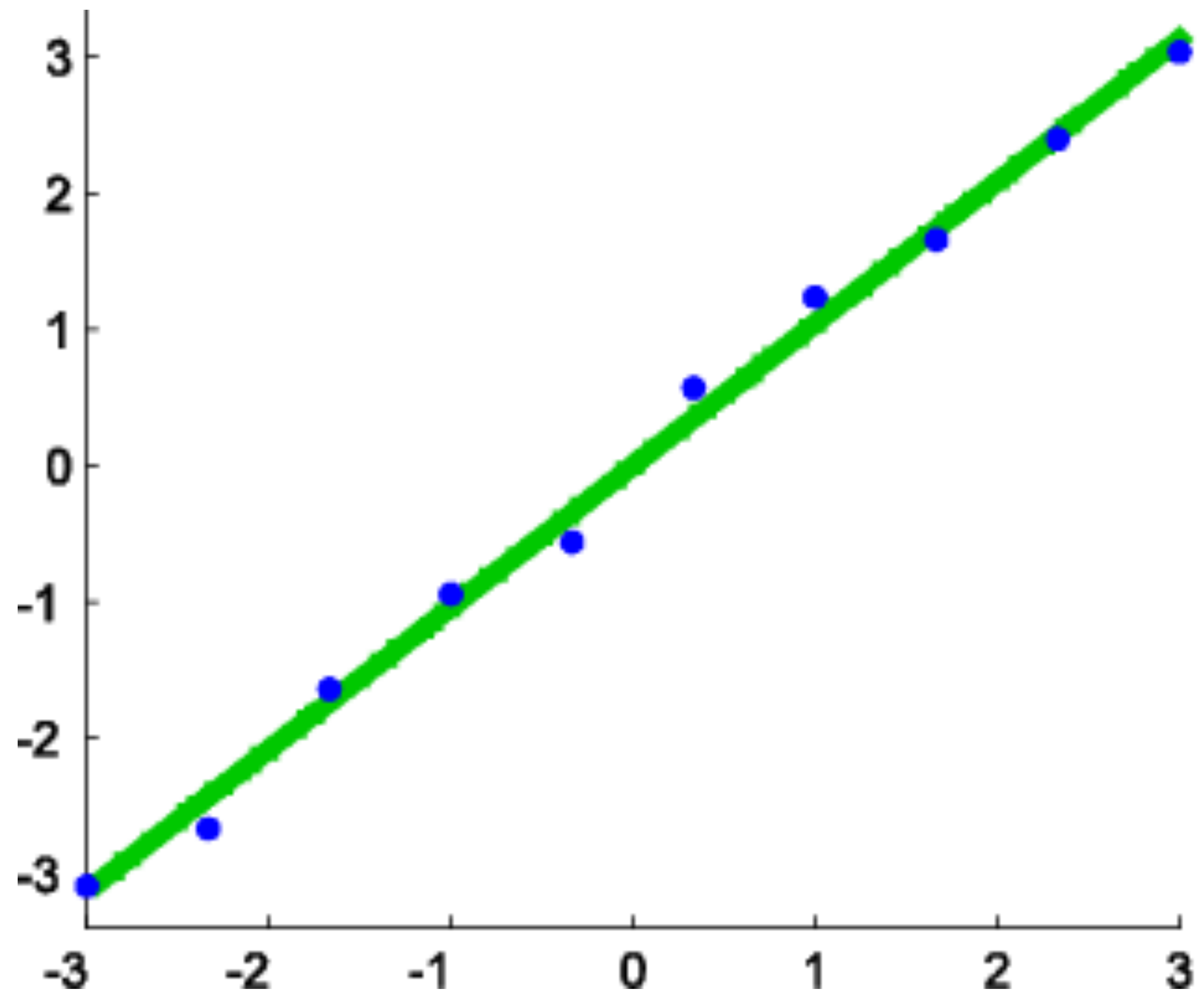
Summary of sparse regression

- Achieve sparsity by ℓ_1-regularized learning.
- We no longer have an analytical solution.
- We can consider many different ideas: fused lasso, group lasso, elastic net, $\cdots$
- Sparse regression is used widely in many domains.

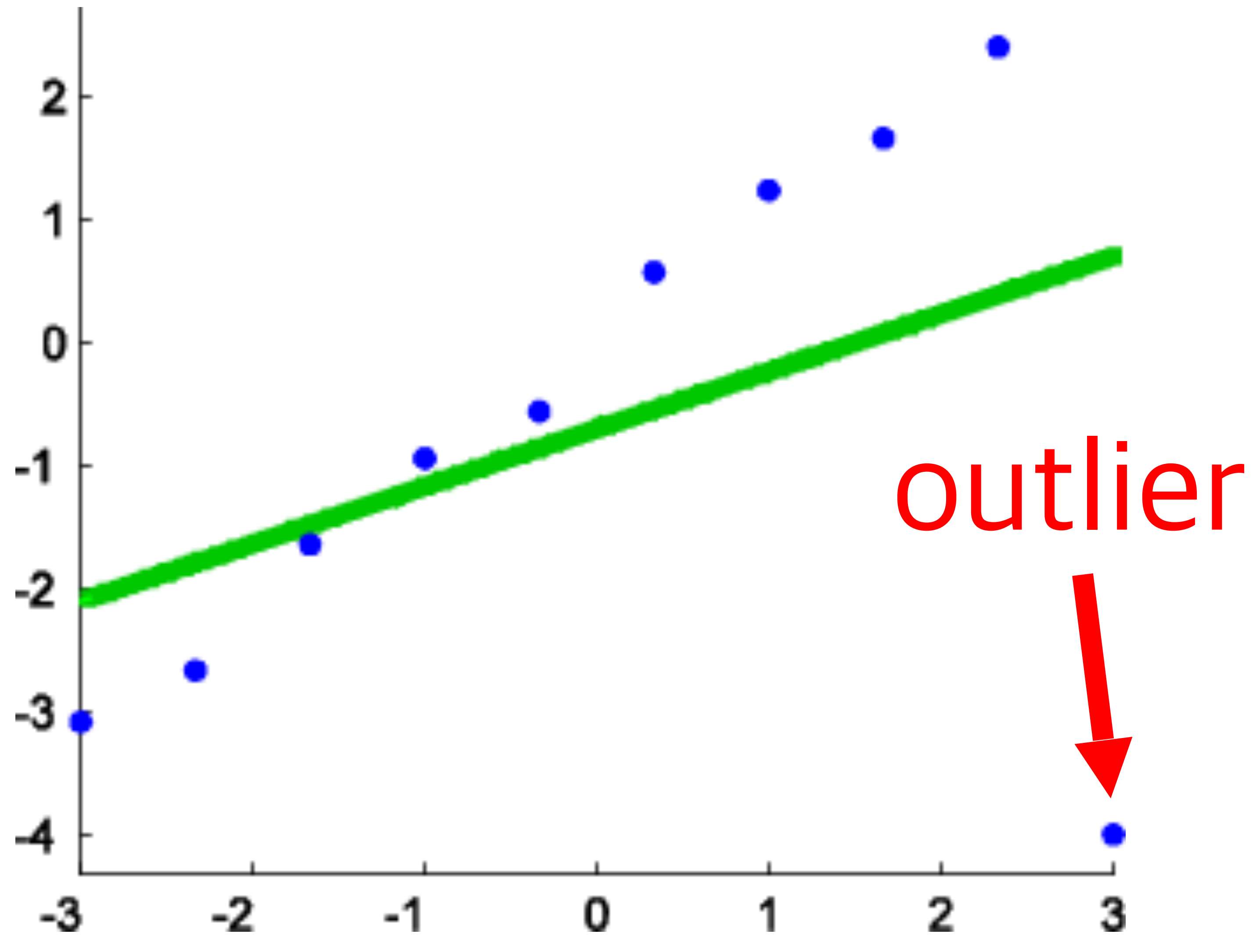
Another Issue of Least Squares

- With just a single outlier, the results are altered heavily:

$$f_{\boldsymbol{\theta}}(x) = \theta_1 + \theta_2 x$$



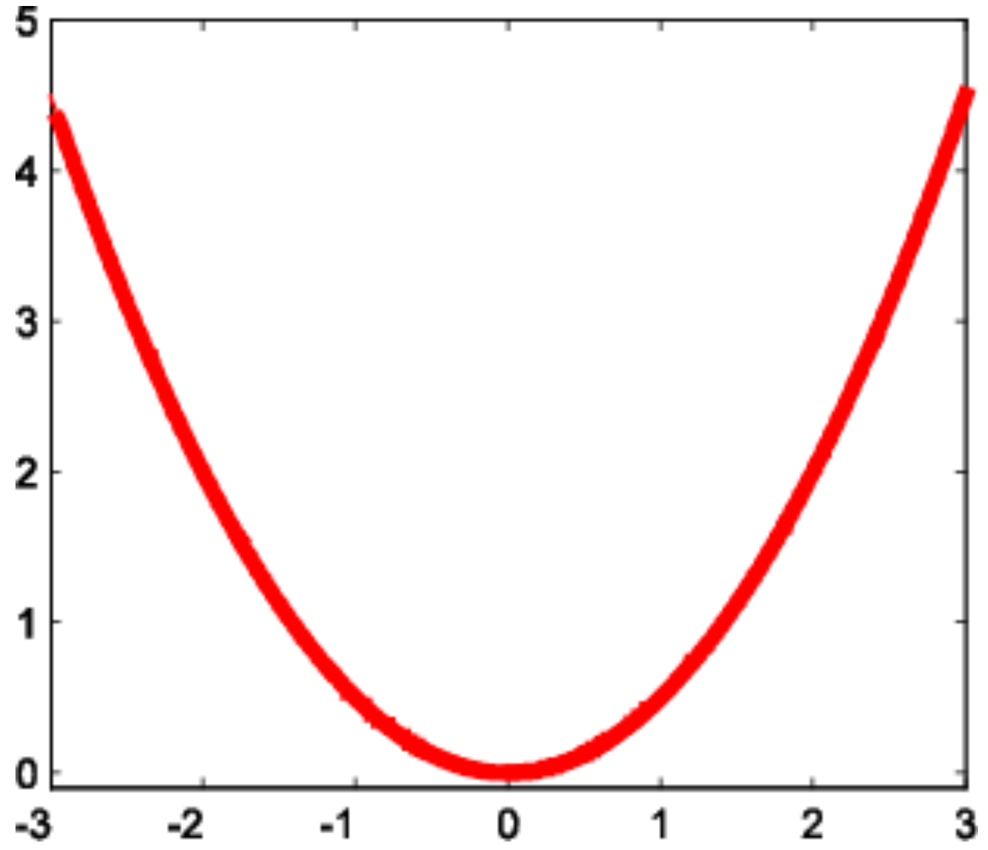
LS regression
(w/o outlier)



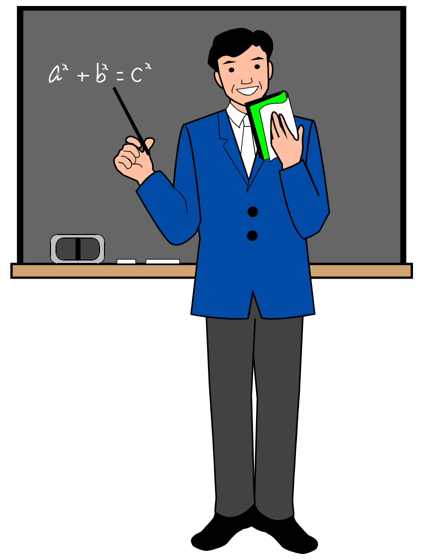
LS regression
(w/ outlier)

ℓ_2 loss function

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



- Least squares regression: measures the goodness of fit of the training output by the ℓ_2 loss function.
- The influence of outliers is strengthened by the “squared” component.
- Need to reduce the influence of outliers to stabilize learning!



Contents

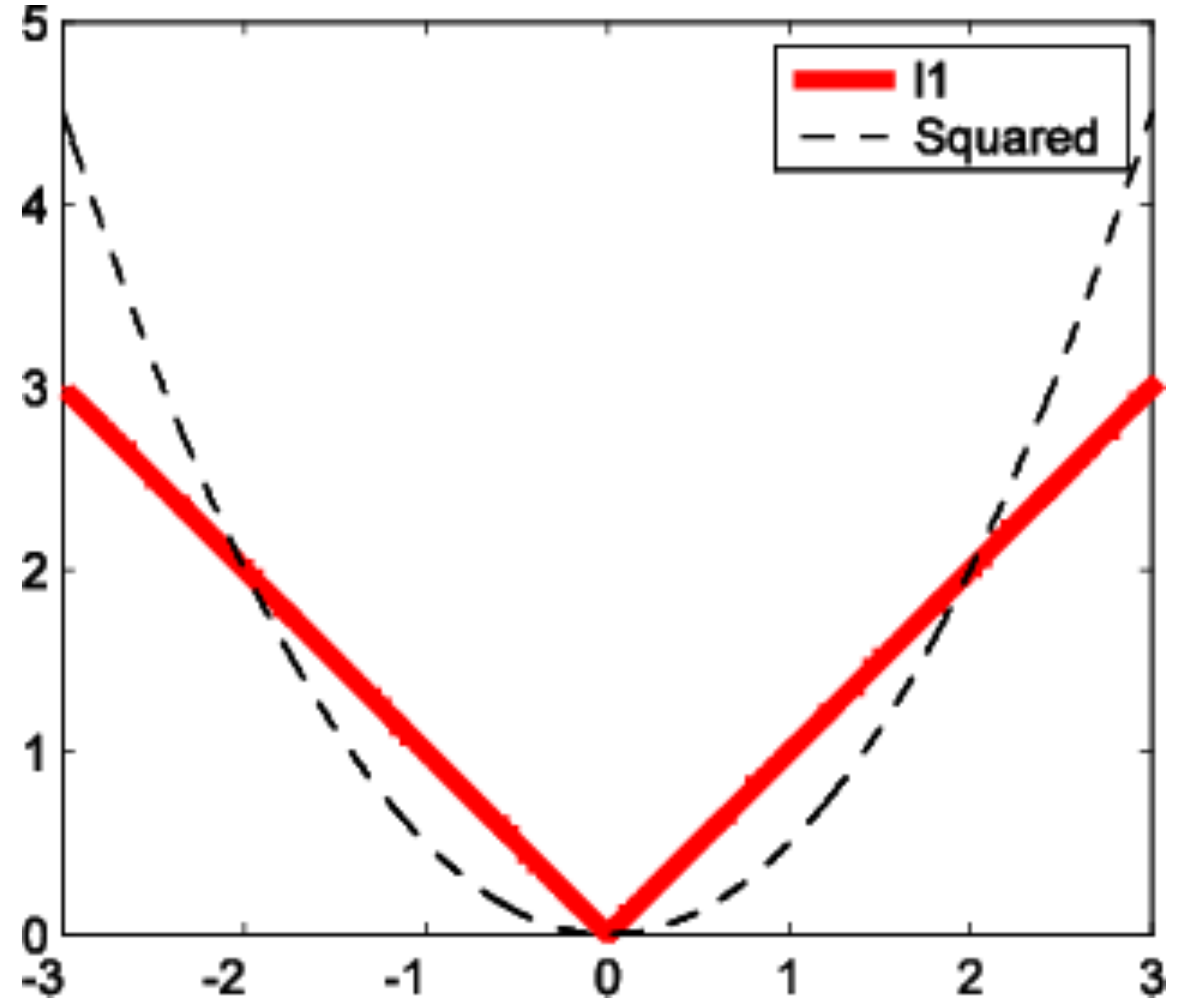
1. Sparse regression
2. Robust regression
   1. ℓ_1-loss
      1. Relationship between median
      2. Robustness and estimation accuracy
   2. Huber loss
   3. Tukey loss

Robust regression with ℓ_1 loss

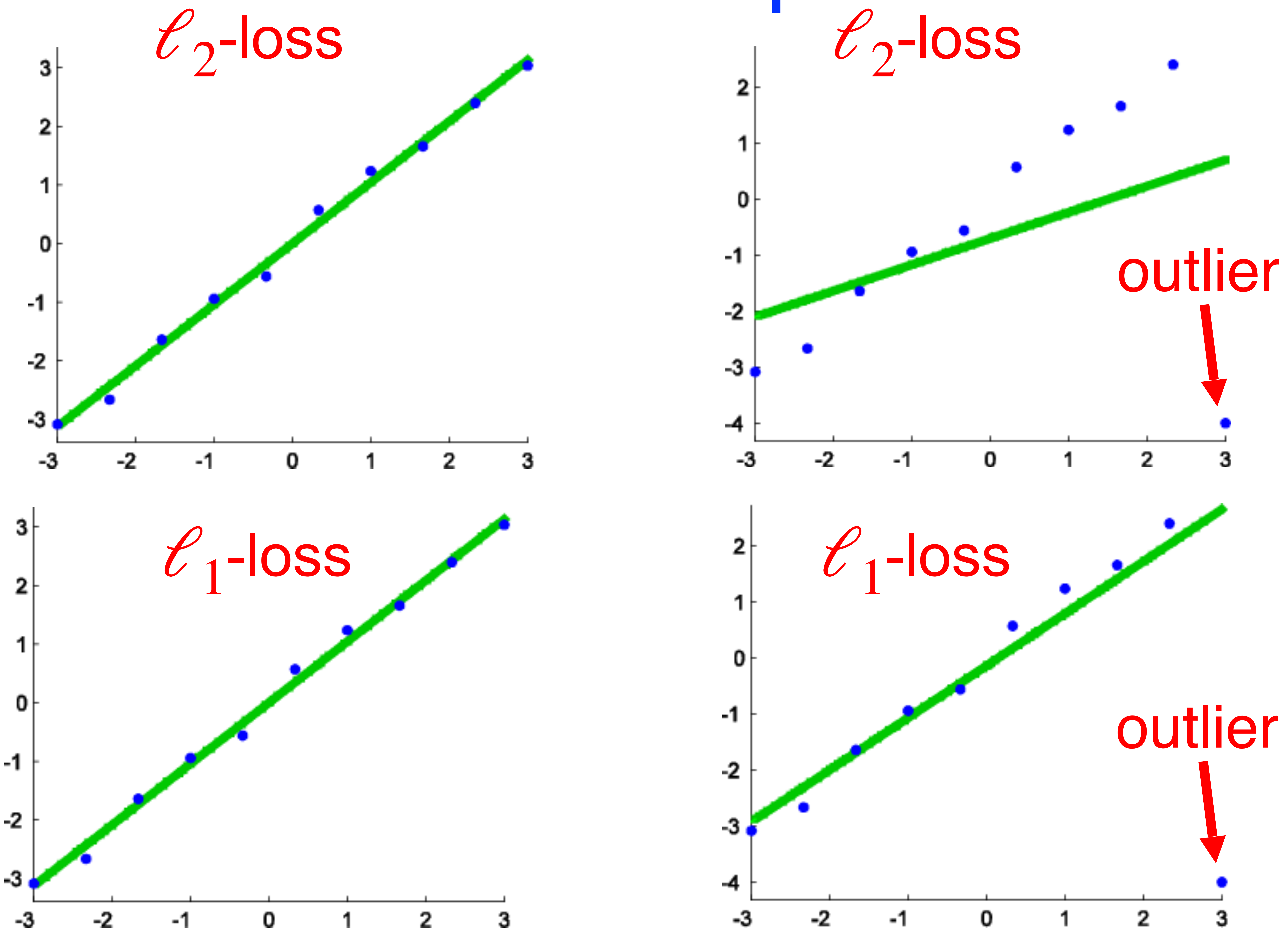
$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left| f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \right|$$

- Goodness of fit to training samples is measured by ℓ_1 function:

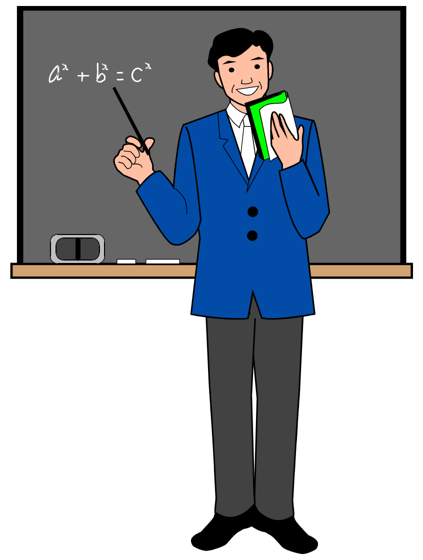
- The influence of the outlier becomes linear.
- Called least absolute (LA).



Example



Will explain the methods soon!



Contents

1. Sparse regression
2. Robust regression
   1. ℓ_1-loss
      1. Relationship between median
      2. Robustness and estimation accuracy
   2. Huber loss
   3. Tukey loss

Cumulative Distribution Function

- Probability of a continuous random variable taking a value less than or equal to x

$$P(x) = \mathrm{Prob}(X \leq x) = \int_{-\infty}^{x} p(u)du$$

$P(x)$: cumulative distribution function

$$P'(x) = \frac{\mathrm{d}P(x)}{\mathrm{d}x} = p(x)$$

- (Weakly) increasing function:

$$x_1 < x_2 \quad \Longrightarrow \quad P(x_1) \leq P(x_2)$$

- Range:

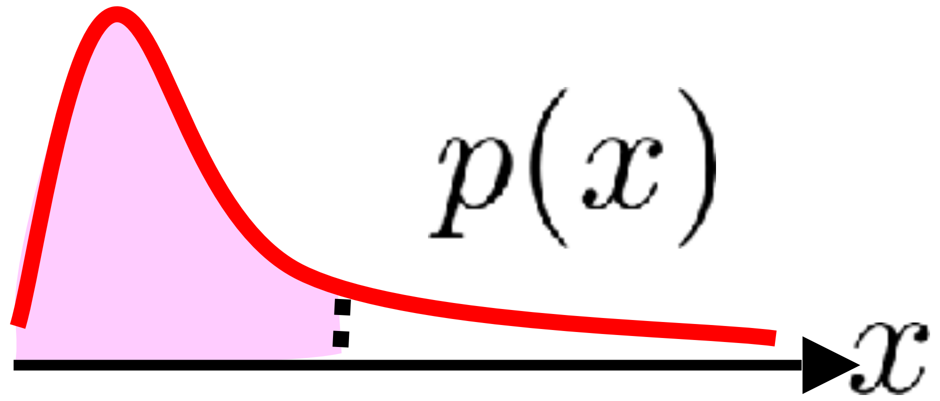
$$x \to -\infty \quad \Longrightarrow \quad P(x) \to 0$$

$$x \to \infty \quad \Longrightarrow \quad P(x) \to 1$$

Expectation and Median

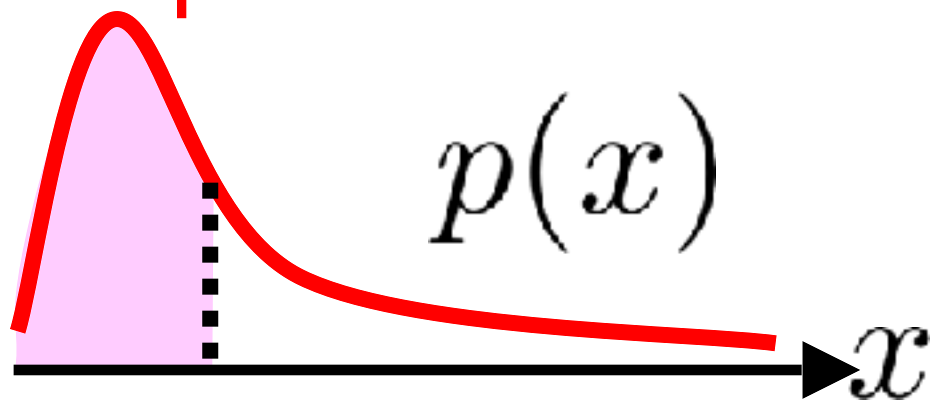
- Expectation:

$$E[X] = \int x p(x) \mathrm{d}x$$

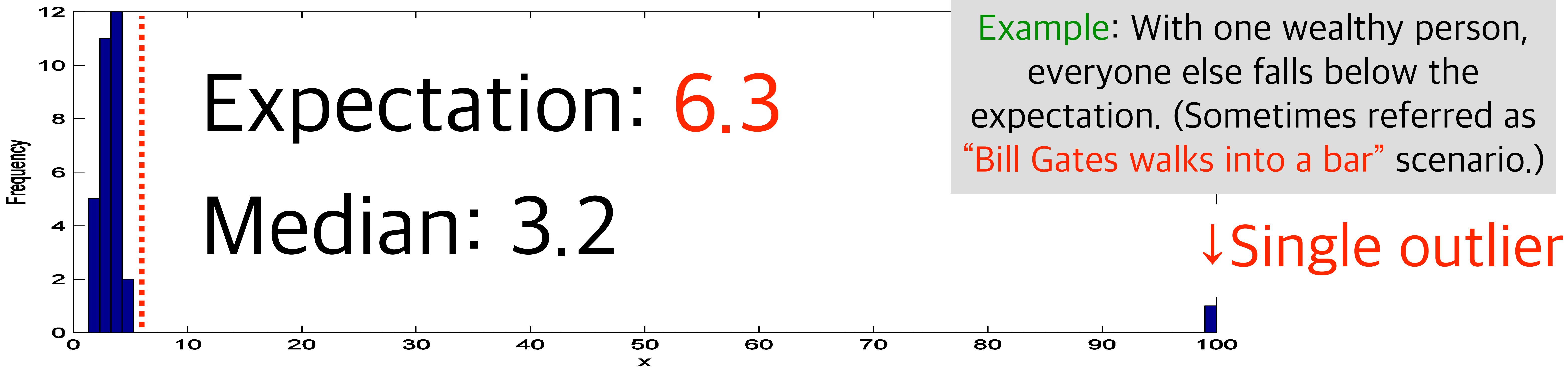


- Median:

$$x \text{ satisfying } \mathrm{Prob}(X \leq x) = \frac{1}{2}$$



- Expected values can be inconsistent with our intuition when there are outliers.



Example: With one wealthy person, everyone else falls below the expectation. (Sometimes referred as "Bill Gates walks into a bar" scenario.)

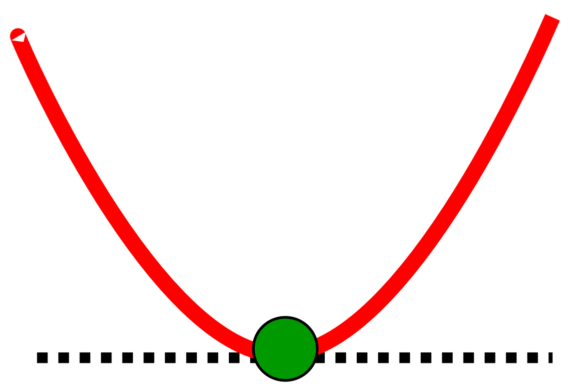
Math exercise

- Consider density function $p(y)$ with support $[a, b]$.
- Show θ_2 that minimizes squared error $J_2(y)$ is the expectation of y.

$$\theta_2 = \underset{\theta}{\operatorname{argmin}}\, J_2(\theta)$$

$$J_2(\theta) = \int_a^b (y - \theta)^2 p(y) dy$$

- Helpful info: derivative is zero at the minimum.



Math exercise

- Consider density function $p(y)$ with support $[a, b]$.
- θ_1 that minimizes the absolute error $J_1(\theta)$ is the median

$$\theta_1 = \underset{\theta}{\operatorname{argmin}}\, J_1(\theta) \qquad J_1(\theta) = \int_a^b |y - \theta| p(y) dy$$

- Helpful info: integration by parts

$$\int_a^b f(y) g'(y) dy = \Big[f(y) g(y) \Big]_a^b - \int_a^b f'(y) g(y) dy$$

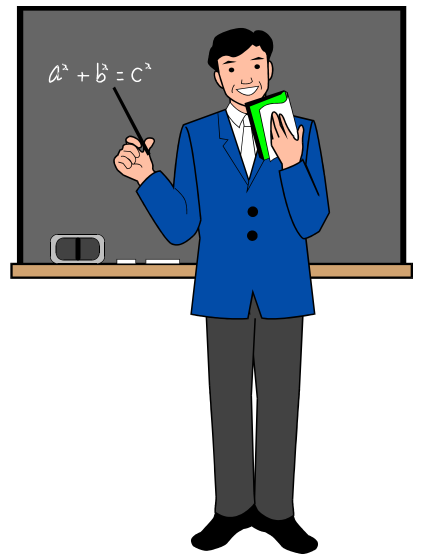
Statistical interpretation of robust learning

- Observed value = true value + noise:

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

- ℓ_2-loss function: corresponds to the estimation of the **expectation** of the observations.
  - $\underset{\mu}{\operatorname{argmin}} \left[\sum_{i=1}^n (y_i - \mu)^2 \right] = \operatorname{mean}\left(\{y_i\}_{i=1}^n \right)$

- ℓ_1-loss function: corresponds to the estimation of **median** of the observations.
  - $\underset{\mu}{\operatorname{argmin}} \left[\sum_{i=1}^n |y_i - \mu| \right] = \operatorname{median}\left(\{y_i\}_{i=1}^n \right)$

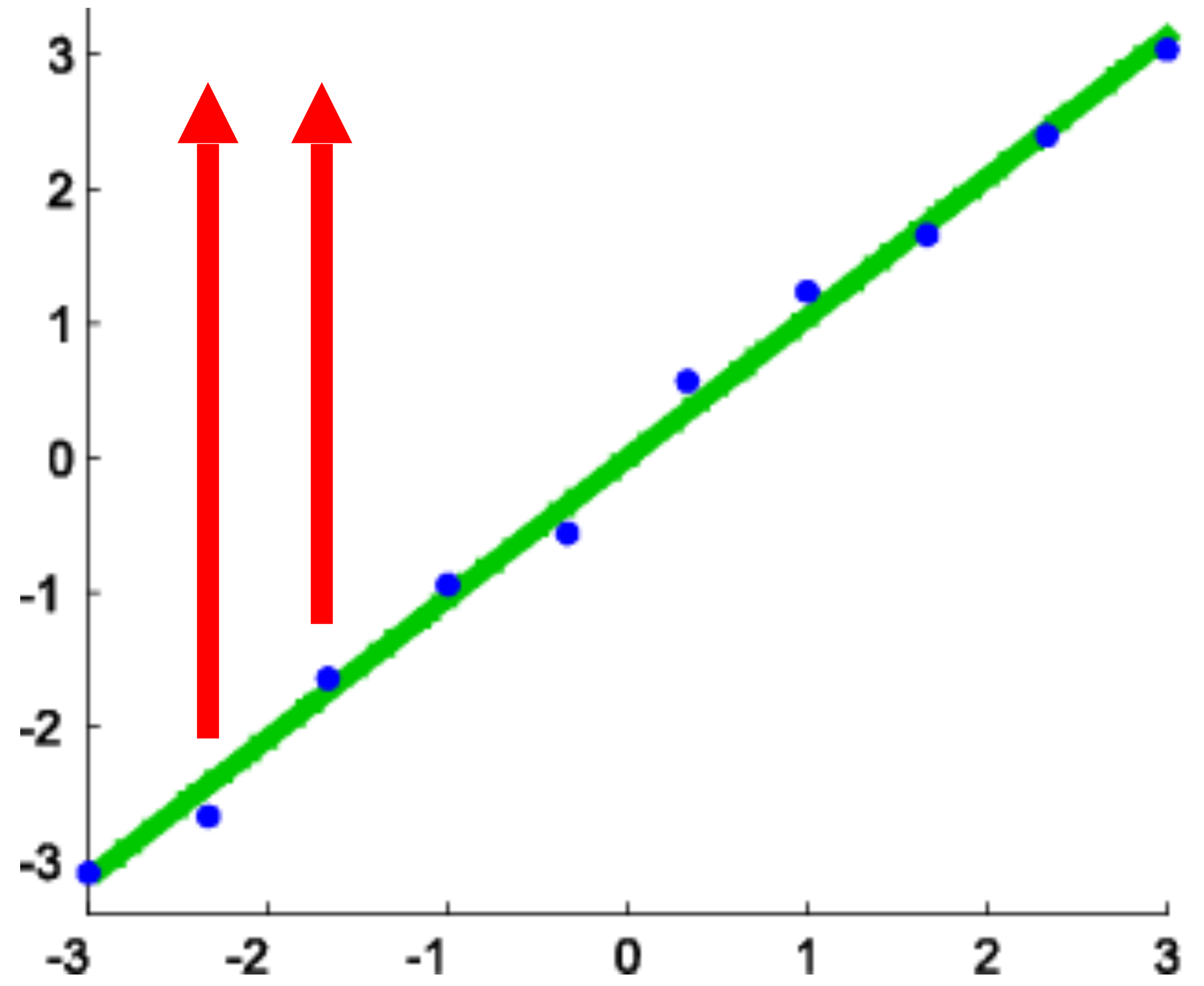


Contents

1. Sparse regression
2. Robust regression
    1. ℓ_1-loss
        1. Relationship between median
        2. Robustness and estimation accuracy
    2. Huber loss
    3. Tukey loss

Robustness and Efficiency

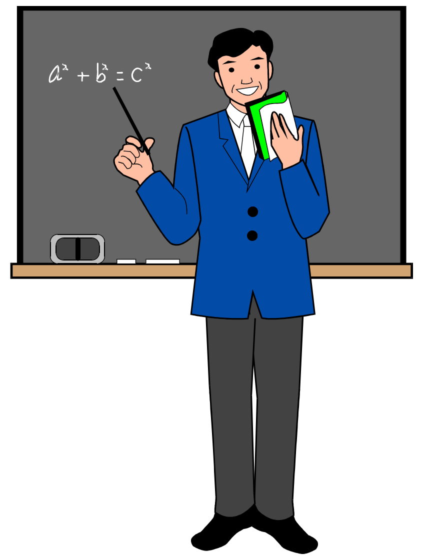
- Breakpoint: the proportion of samples that will maintain non-breaking solutions when we replace samples with infinity.
  - ℓ_2-loss function: 0% (🚨 not robust)
  - ℓ_1-loss function: 50% (😎 robust)



- However, the ℓ_1-loss function is not efficient for Gaussian noise (large variance).

Requirements for robust regression

- Highly robust = ignoring more training samples
  - A regressor that always outputs 0 is meaningless but super robust.
- Practical requirements:
  - Want to be similar to LS when there are no outliers.
  - Need robustness when there are many or large outliers.



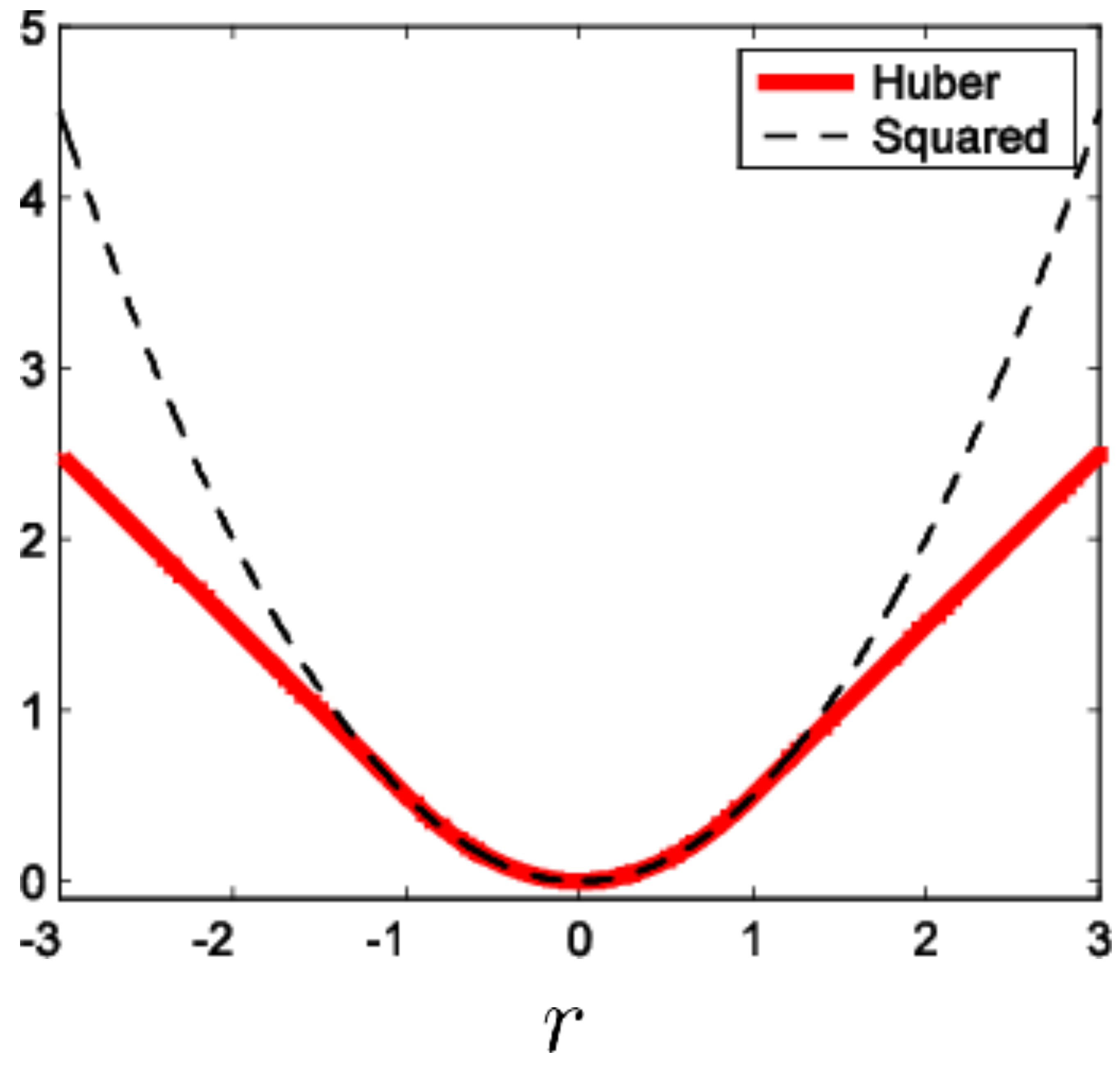
Contents

1. Sparse regression
2. Robust regression
   1. ℓ_1-loss
   2. Huber loss
   3. Tukey loss

Huber loss

- The best of both world?
  - Squared for small errors
  - Absolute for larger errors

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \rho(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i)$$



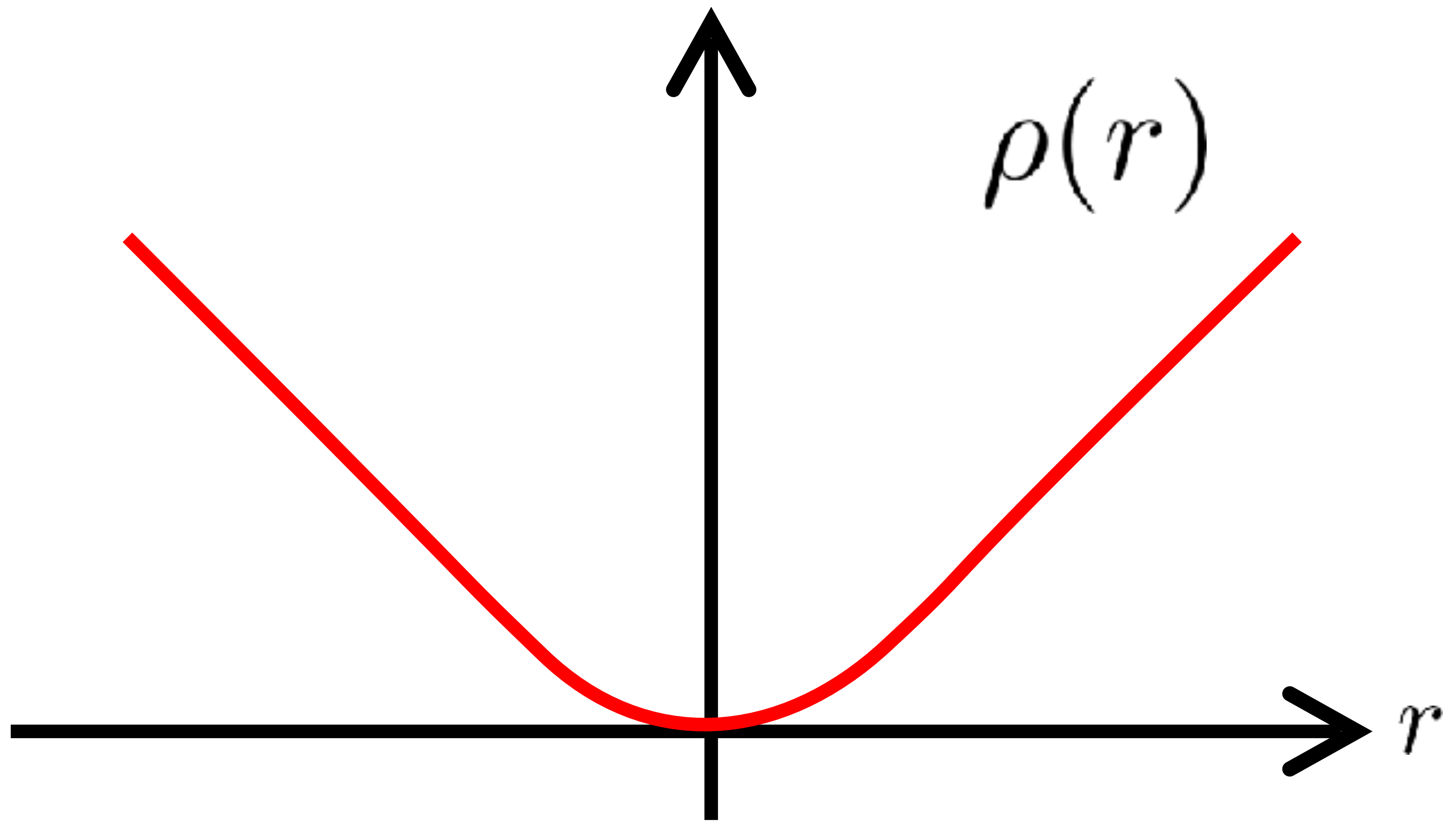
$$\rho(r) = \begin{cases} r^2/2 & (|r| \leq \eta) \\ \eta|r| - \eta^2/2 & (|r| > \eta) \end{cases}$$

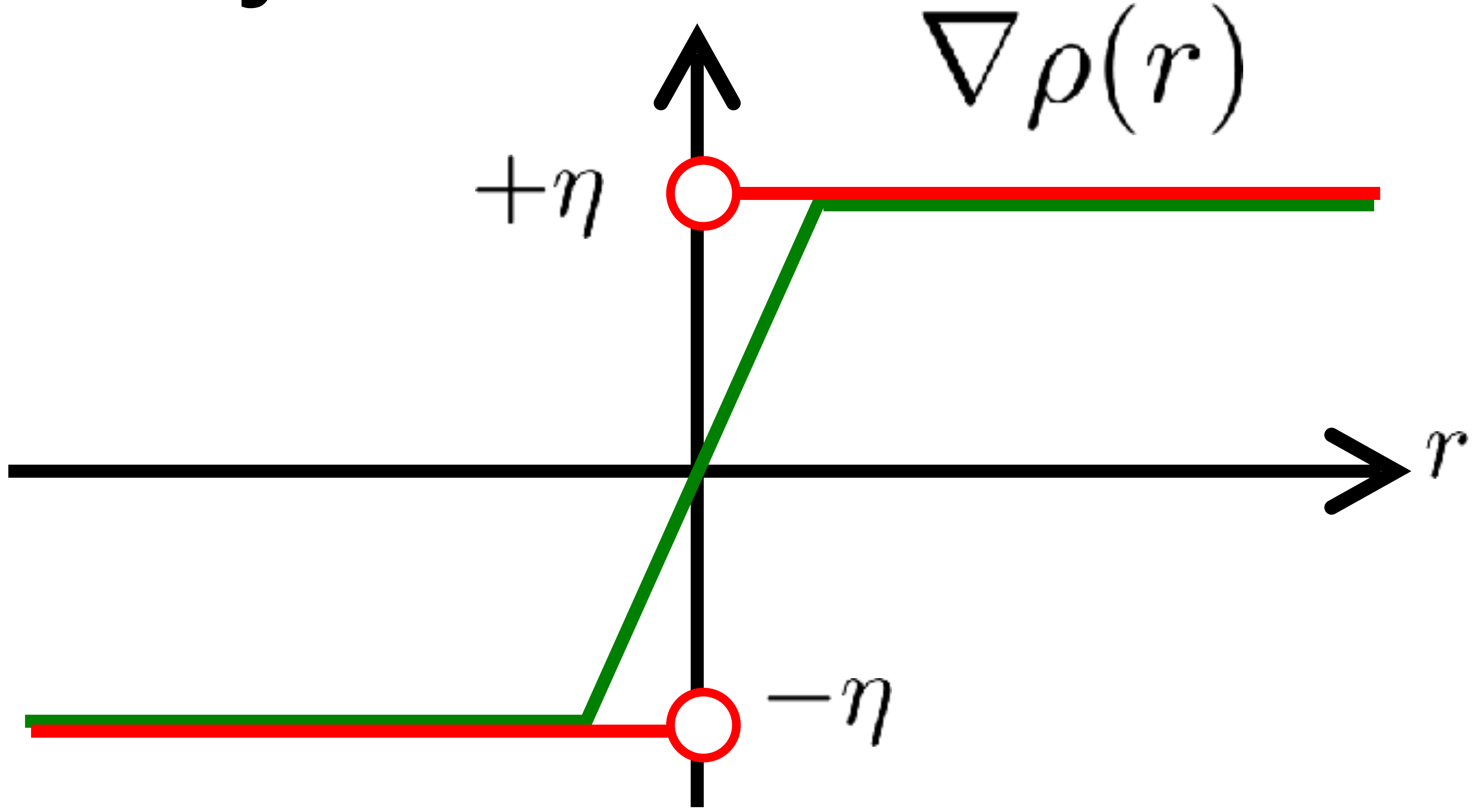
$$\eta \geq 0$$

- Parameter η is designed by the user as the point at which errors may originate from outliers.

1st Approach

- Huber loss is continuously differentiable.





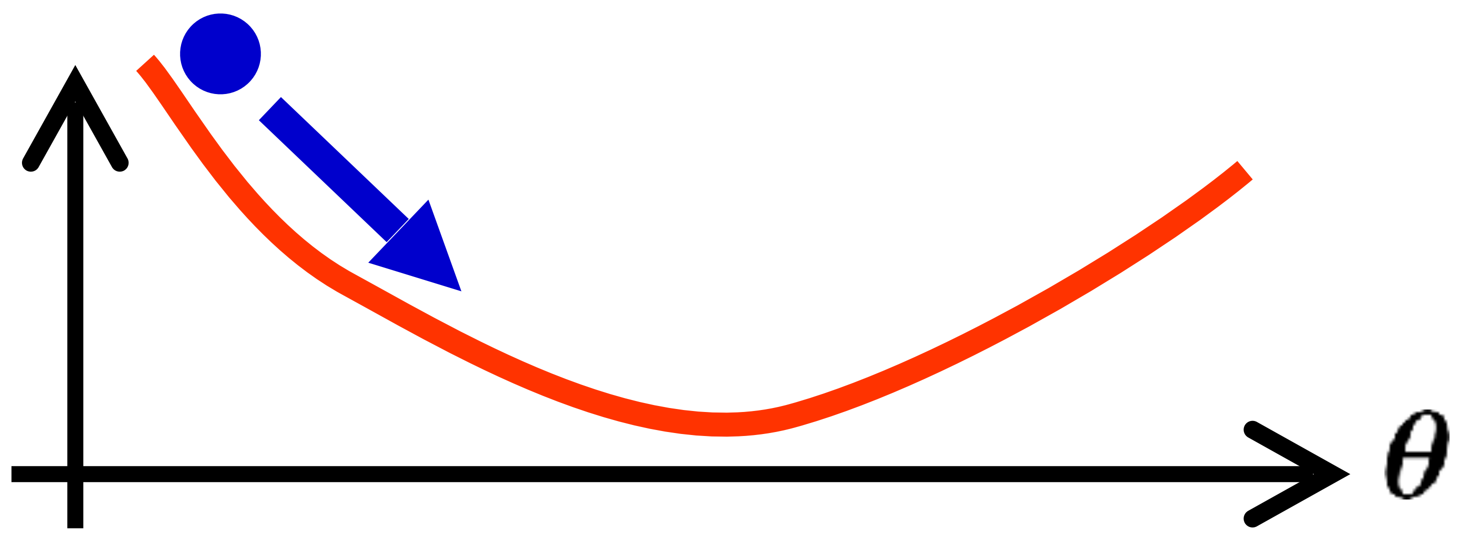
$$\rho(r) = \begin{cases} r^2/2 & (|r| \leq \eta) \\ \eta|r| - \eta^2/2 & (|r| > \eta) \end{cases}$$

$$\rho'(r) = \begin{cases} r & (|r| \leq \eta) \\ \mathrm{sign}(r)\eta & (|r| > \eta) \end{cases}$$

- Gradient method:

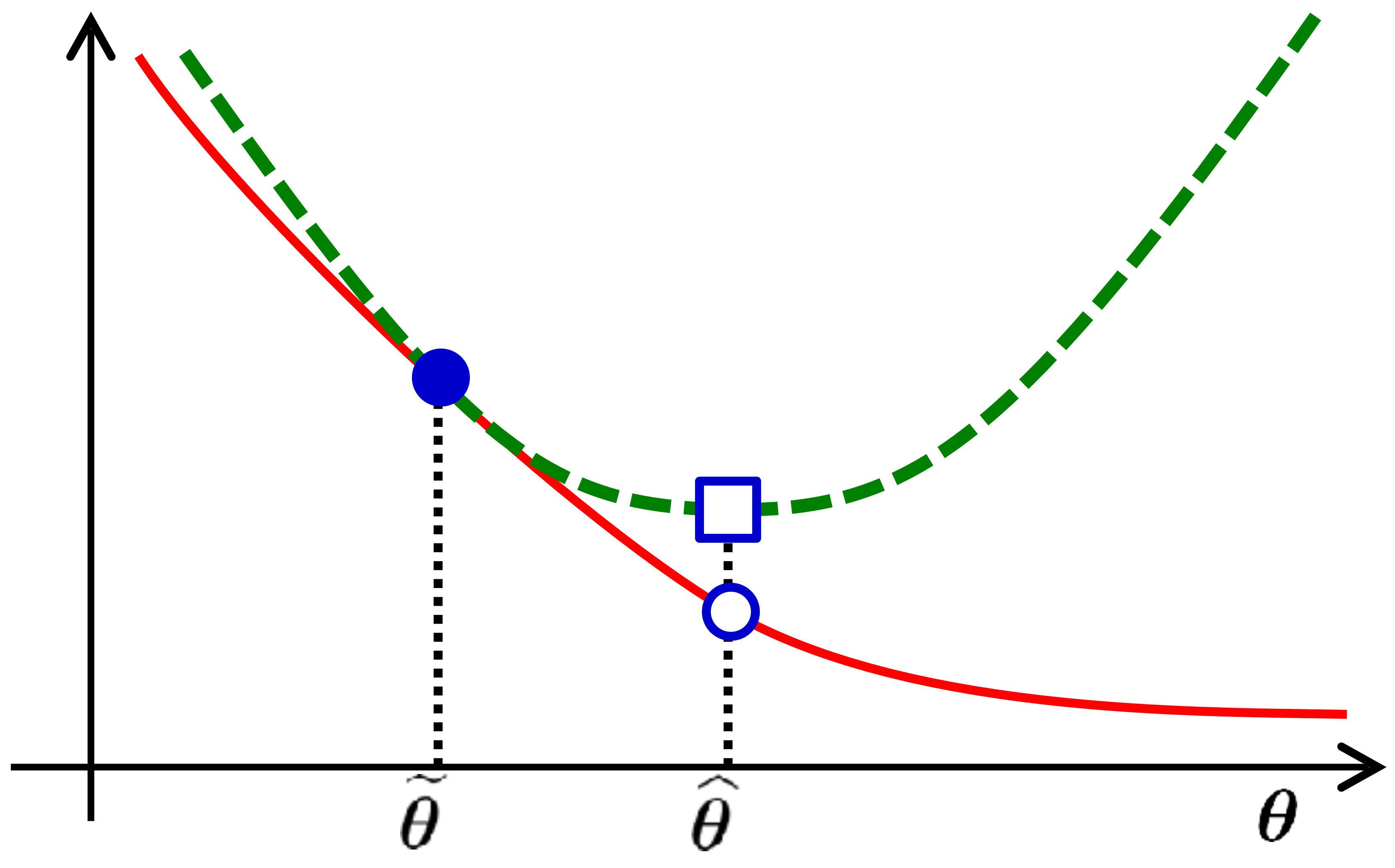
$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} - \varepsilon \nabla J(\boldsymbol{\theta})$$

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \rho(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i)$$



2nd Approach

- Gradient methods are tricky to adjust step widths
- Iterative least squares algorithm:
  - Huber loss is suppressed from above by a quadratic function tangent to the current solution (different from Newton's method)
  - Minimizing the quadratic upper bound analytically to find a better solution step by step



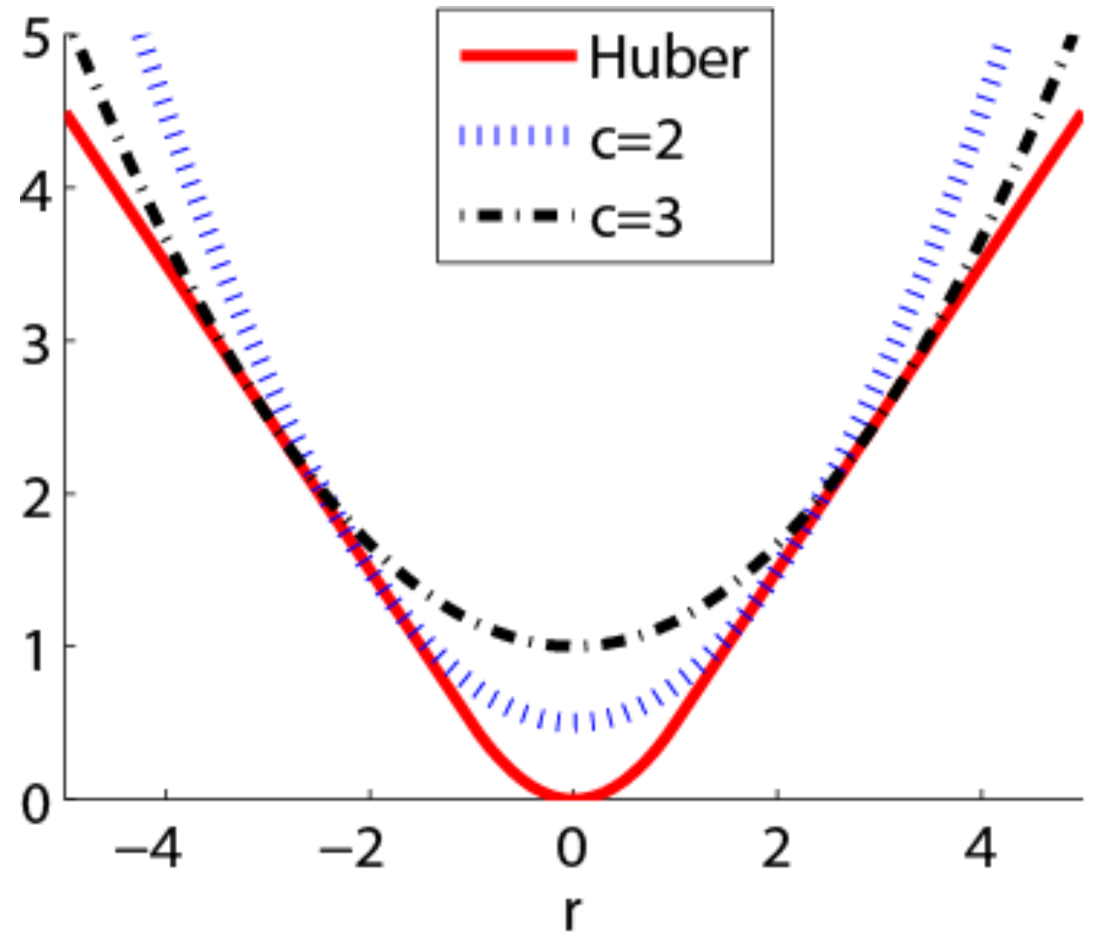
Math exercise

- Huber loss when $|r| > \eta$: $\eta|r| - \dfrac{\eta^2}{2}$

$$\rho(r) = \begin{cases} r^2/2 & (|r| \le \eta) \\ \eta|r| - \eta^2/2 & (|r| > \eta) \end{cases}$$

  Derive a quadratic function that is tangent to the huber loss.

- Helpful info: from symmetry, quadratic function that is tangent at $\pm c$ can be expressed as $ar^2 + b$



Minimization of upper bound

$$\rho(r) = \begin{cases} r^2/2 & (|r| \leq \eta) \\ \eta|r| - \eta^2/2 & (|r| > \eta) \end{cases}$$

- Upper bound $\tilde{\rho}(r) \geq \rho(r)$ for residual $\tilde{r} = f_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{x}) - y$ (where $\tilde{\boldsymbol{\theta}}$ is current param):

$$\widetilde{\rho}(r) = \begin{cases} r^2/2 & (|\widetilde{r}| \leq \eta) \\ \frac{\eta}{2|\widetilde{r}|} r^2 + \underbrace{\frac{\eta|\widetilde{r}|}{2} - \frac{\eta^2}{2}}_{\text{constant}} & (|\widetilde{r}| > \eta) \end{cases}$$

$$= \frac{\widetilde{w}}{2} r^2 + \text{const} \qquad \widetilde{w} = \begin{cases} 1 & (|\widetilde{r}| \leq \eta) \\ \eta/|\widetilde{r}| & (|\widetilde{r}| > \eta) \end{cases}$$

Minimization of upper bound (cont.)

- Original objective we wanted to minimize:

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \rho\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\right) \qquad \rho(r) = \begin{cases} r^2/2 & (|r| \le \eta) \\ \eta|r| - \eta^2/2 & (|r| > \eta) \end{cases}$$

- Minimization of $\widetilde{J}$ (upper bound of J) derived with $\tilde{\boldsymbol{\theta}}$:

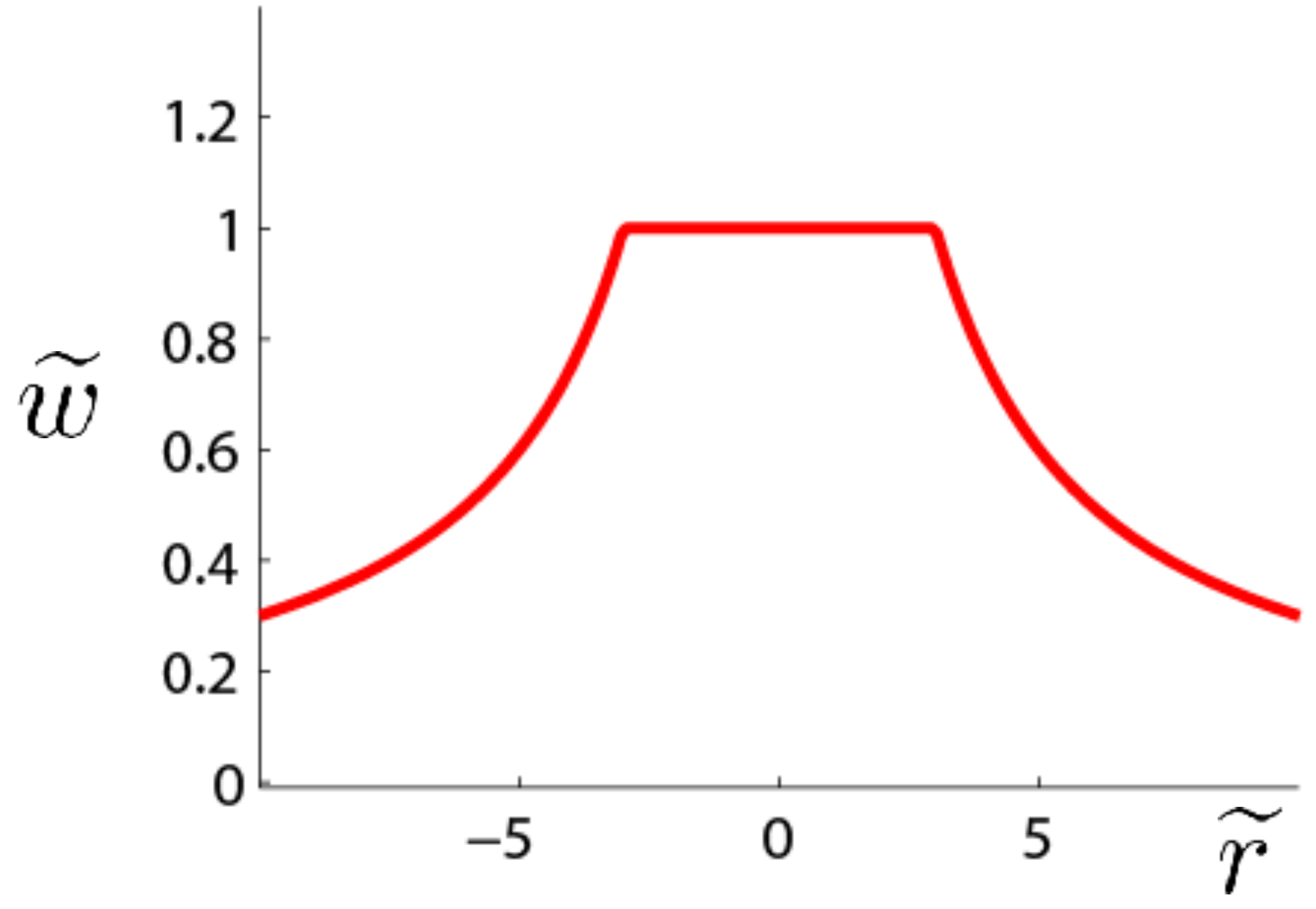
$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \widetilde{J}(\boldsymbol{\theta}) \qquad \widetilde{J}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n} \widetilde{w}_i \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i\right)^2$$

$$\widetilde{w}_i = \begin{cases} 1 & (|\widetilde{r}_i| \le \eta) \\ \eta/|\widetilde{r}_i| & (|\widetilde{r}_i| > \eta) \end{cases} \qquad \widetilde{r}_i = f_{\widetilde{\boldsymbol{\theta}}}(\boldsymbol{x}_i) - y_i$$

Minimization of upper bound (cont.)

- Upper bound is weighted least squares:

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} \widetilde{w}_i \Big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \Big)^2$$



- Small weights are set for outliers

$$\widetilde{w}_i = \begin{cases} 1 & (|\widetilde{r}_i| \leq \eta) \\ \eta / |\widetilde{r}_i| & (|\widetilde{r}_i| > \eta) \end{cases} \qquad \widetilde{r}_i = f_{\widetilde{\boldsymbol{\theta}}}(\boldsymbol{x}_i) - y_i$$

- The minimum solution of the upper bound is analytically obtained by $\widehat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^\top \widetilde{\boldsymbol{W}} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \widetilde{\boldsymbol{W}} \boldsymbol{y}$

$$\widetilde{\boldsymbol{W}} = \mathrm{diag}\left(\widetilde{w}_1, \ldots, \widetilde{w}_n\right)$$

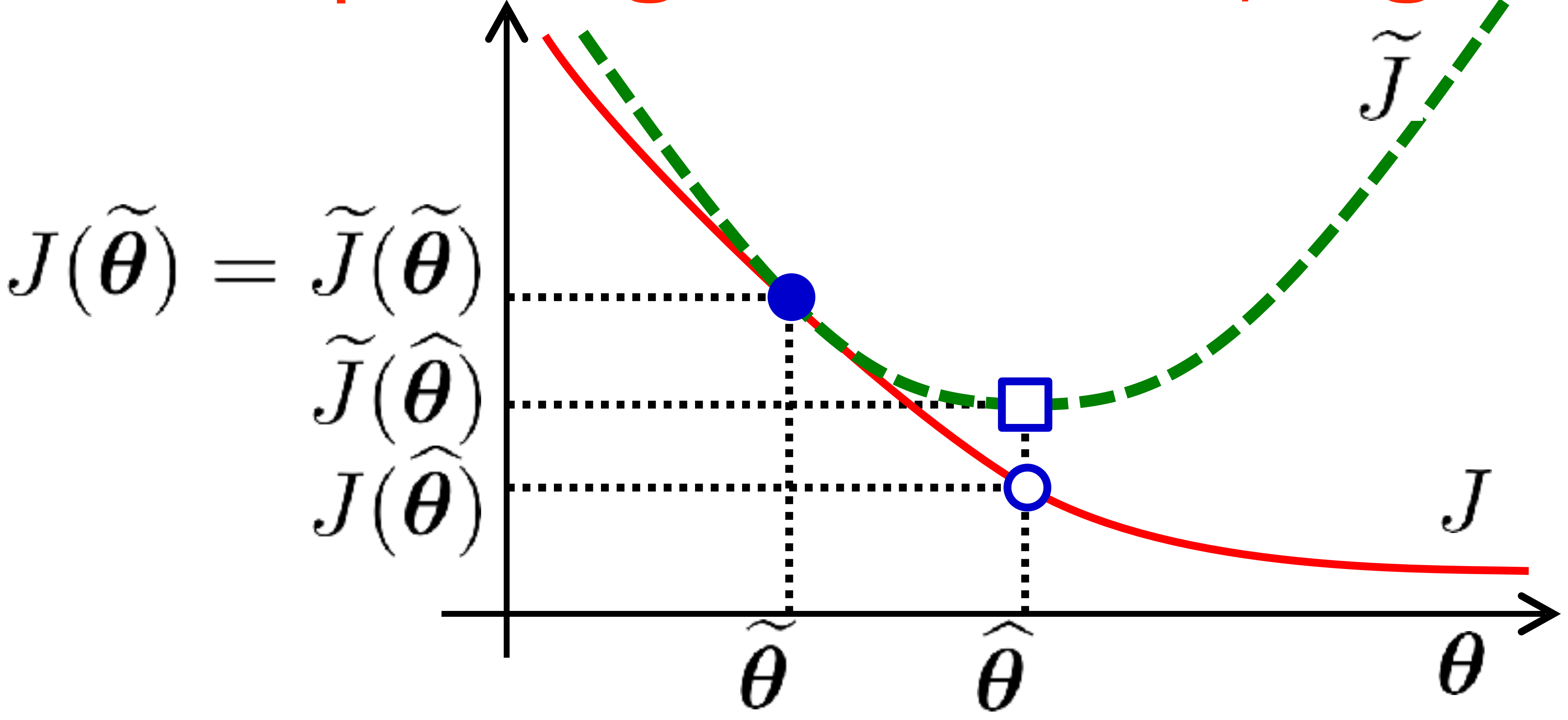
Proving this is homework.

Minimization of upper bound (cont.)

- Since the upper bound is tangent at $\tilde{\boldsymbol{\theta}}$: $J(\widetilde{\boldsymbol{\theta}}) = \widetilde{J}(\widetilde{\boldsymbol{\theta}})$
- If $\hat{\boldsymbol{\theta}}$ is the minimal solution of the upper bound: $\widetilde{J}(\widetilde{\boldsymbol{\theta}}) \geq \widetilde{J}(\widehat{\boldsymbol{\theta}})$

- Since $\tilde{J}$ is the upper bound of J: $\widetilde{J}(\widehat{\boldsymbol{\theta}}) \geq J(\widehat{\boldsymbol{\theta}})$
- To summarize: $J(\widetilde{\boldsymbol{\theta}}) = \widetilde{J}(\widetilde{\boldsymbol{\theta}}) \geq \widetilde{J}(\widehat{\boldsymbol{\theta}}) \geq J(\widehat{\boldsymbol{\theta}})$

💡 when updating from $\tilde{\boldsymbol{\theta}}$ to $\hat{\boldsymbol{\theta}}$, J generally decreases



$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \widetilde{J}(\boldsymbol{\theta})$$

Iteratively reweighted LS algorithm

- Initialize $\boldsymbol{\theta}$.
- Repeat below until convergence:
  - Derive $\boldsymbol{W}$ from current solution $\boldsymbol{\theta}$ (derive upper bound)

$$\boldsymbol{W} = \mathrm{diag}\left(w_1, \ldots, w_n\right)$$

$$w_i = \begin{cases} 1 & (|f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i| \leq \eta) \\ \eta / |f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i| & (|f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i| > \eta) \end{cases}$$

  - Update $\boldsymbol{\theta}$ (minimize upper bound)

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

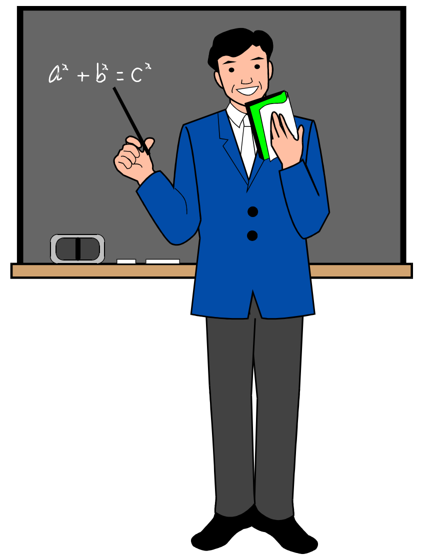
Python implementation 1

- Iterative least squares algorithm for Huber regression for the linear model $f_{\theta}(x) = \theta_1 + \theta_2 x$.

```
import numpy as np
import matplotlib

matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

np.random.seed(1)

def generate_sample(x_min=-3., x_max=3., sample_size=10):
    x = np.linspace(x_min, x_max, num=sample_size)
    y = x + np.random.normal(loc=0., scale=.2, size=sample_size)
    y[-1] = -4  # outlier
    return x, y

def build_design_matrix(x):
    phi = np.empty(x.shape + (2,))
    phi[:, 0] = 1.
    phi[:, 1] = x
    return phi

def iterative_reweighted_least_squares(x, y, eta=1., n_iter=1000):
    phi = build_design_matrix(x)
    # initialize theta using the solution of regularized ridge regression
    theta = theta_prev = np.linalg.solve(
        phi.T.dot(phi) + 1e-4 * np.identity(phi.shape[1]), phi.T.dot(y))
    for _ in range(n_iter):
        r = np.abs(phi.dot(theta_prev) - y)
        w = np.diag(np.where(r > eta, eta / r, 1.))
        phit_w_phi = phi.T.dot(w).dot(phi)
        phit_w_y = phi.T.dot(w).dot(y)
        theta = np.linalg.solve(phit_w_phi, phit_w_y)
        if np.linalg.norm(theta - theta_prev) < 1e-3:
            break
        theta_prev = theta
    return theta
```

Python implementation 2


```
def predict(x, theta):
    phi = build_design_matrix(x)
    return phi.dot(theta)

def visualize(x, y, theta, x_min=-4., x_max=4., filename='xxxxxx.png'):
    X = np.linspace(x_min, x_max, 1000)
    Y = predict(X, theta)
    plt.clf()
    plt.plot(X, Y, color='green')
    plt.scatter(x, y, c='blue', marker='o')
    plt.savefig(filename)

x, y = generate_sample()
theta = iterative_reweighted_least_squares(x, y, eta=1.)
visualize(x, y, theta)
```



Contents

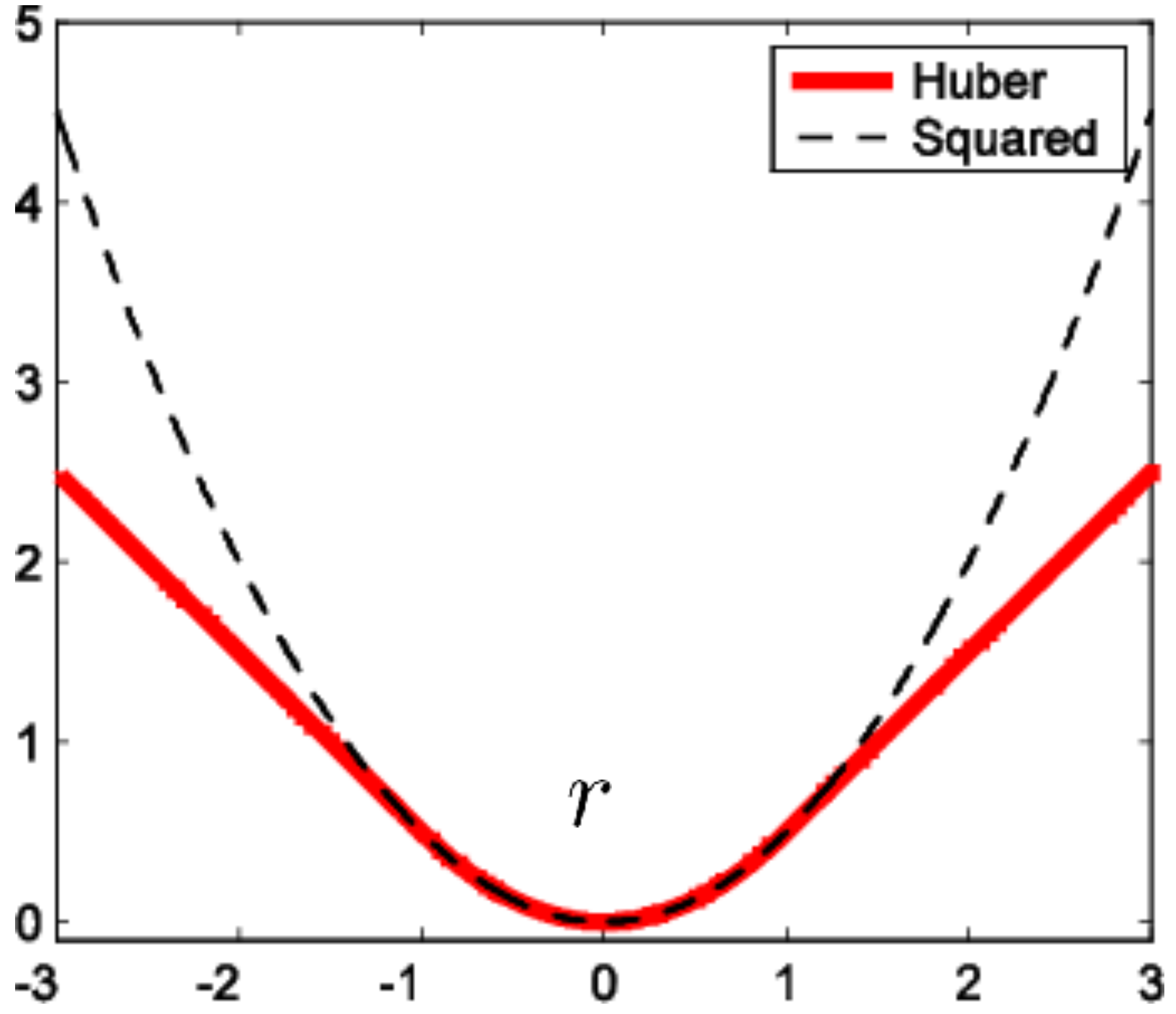
1. Sparse regression
2. Robust regression
    1. ℓ_1-loss
    2. Huber loss
    3. Tukey loss

Can we improve further?

- Huber loss is robust compared with ℓ_2-loss
- But… no upper bound on the loss, so the effect of outliers remain to some extent.

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \rho_{\text{Huber}}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i)$$

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

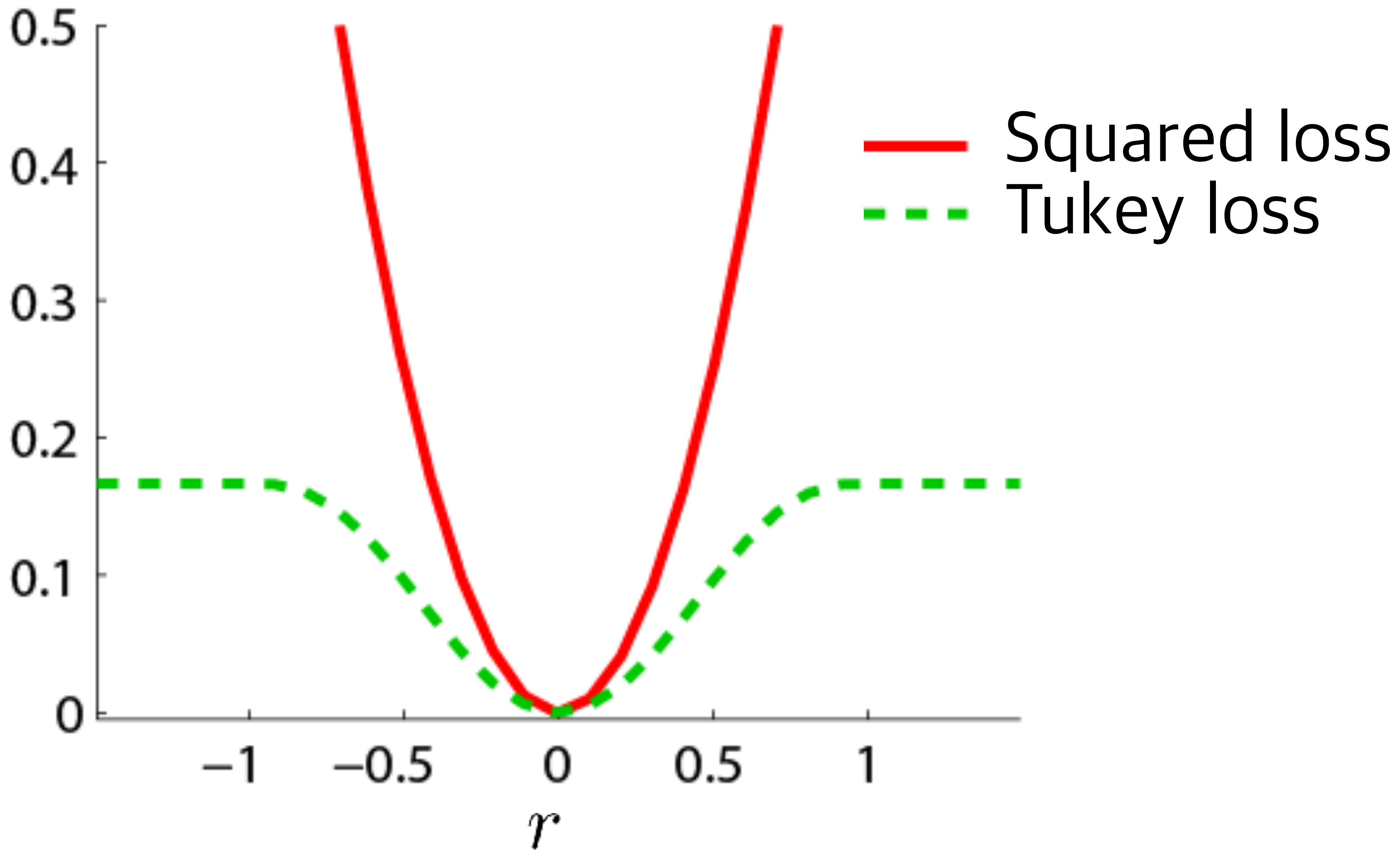


Tukey loss

$$r = f_{\boldsymbol{\theta}}(\boldsymbol{x}) - y$$

- Consider an upper bounded loss

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



Minimization of quadratic upper bound for general losses

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \rho(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i)$$

- Consider differentiable and symmetric loss $\rho(r)$.
- Quadratic upper bound that is tangent to $\rho(r)$ at $\pm\tilde{r}$:

$$\widetilde{\rho}(r) = \frac{\widetilde{w}}{2} r^2 + \text{const}$$

$\widetilde{w} = \rho'(\widetilde{r})/\widetilde{r}$

(You can try to prove this yourself)

- Iterative LS algorithm:

$$\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} \widetilde{w}_i \Big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \Big)^2$$

$\widetilde{w}_i = \rho'(\widetilde{r}_i)/\widetilde{r}_i$

$\widetilde{r}_i = f_{\widetilde{\boldsymbol{\theta}}}(\boldsymbol{x}_i) - y_i$

Weight function for Tukey loss

- Tukey loss:

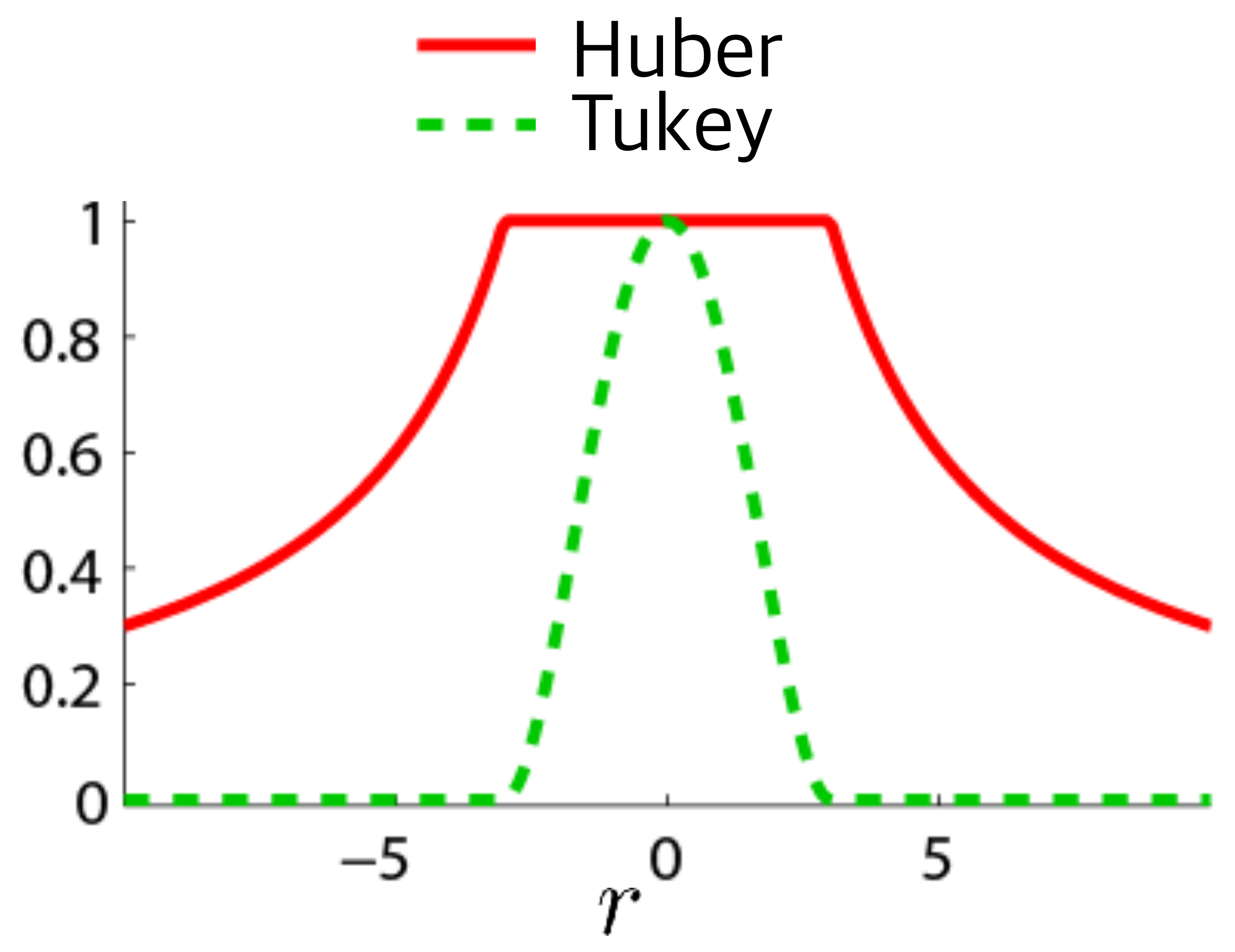
$$\rho_{\text{Tukey}}(r) = \begin{cases} \left(1 - \left[1 - r^2/\eta^2\right]^3\right)/6 & (|r| \leq \eta) \\ 1/6 & (|r| > \eta) \end{cases}$$

- Weight function for Tukey loss: $\widetilde{w} = \rho'(\widetilde{r})/\widetilde{r}$

$$w_{\text{Tukey}} = \begin{cases} \left(1 - r^2/\eta^2\right)^2 & (|r| \leq \eta) \\ 0 & (|r| > \eta) \end{cases}$$

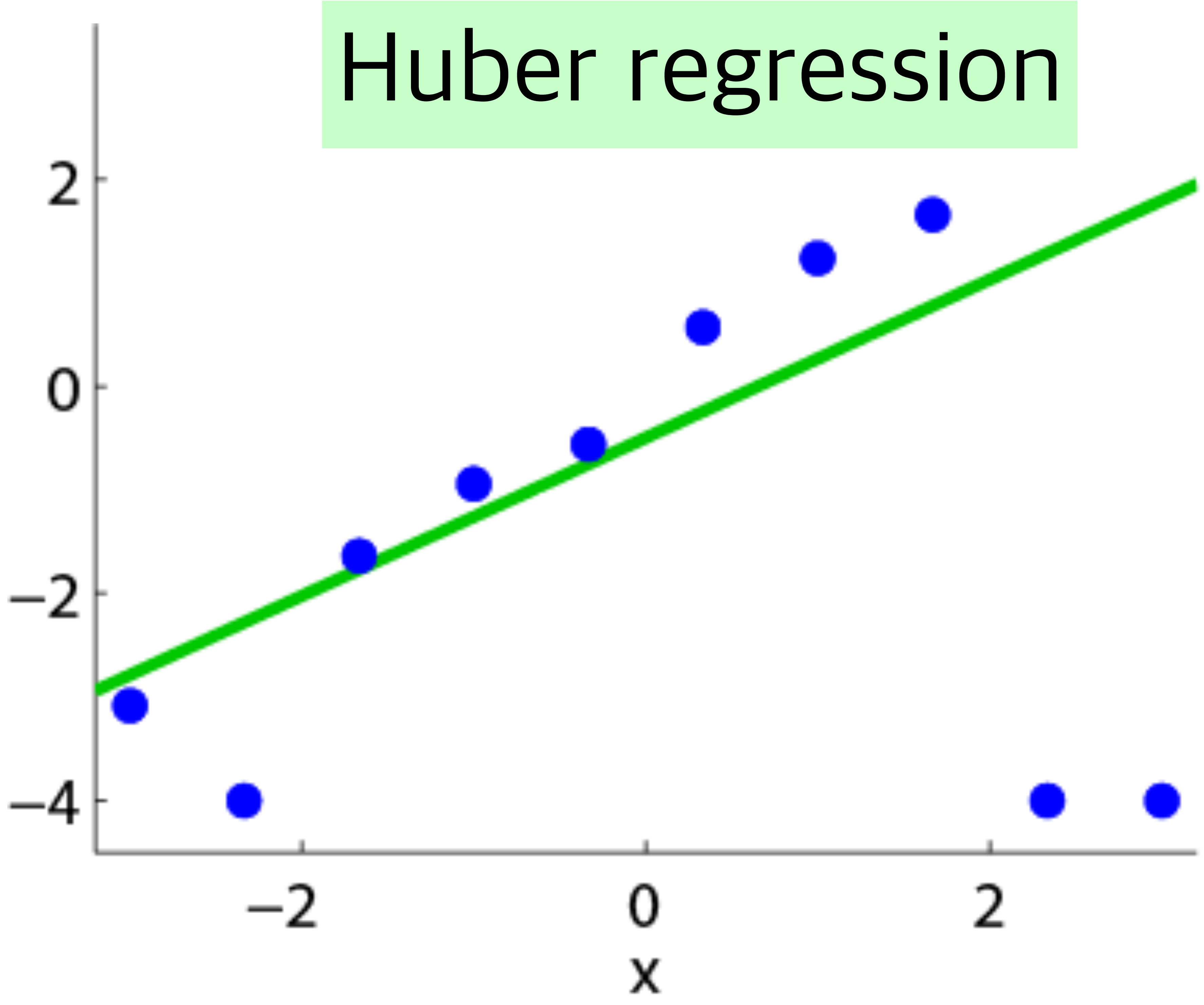
  - Weight is zero for large residuals

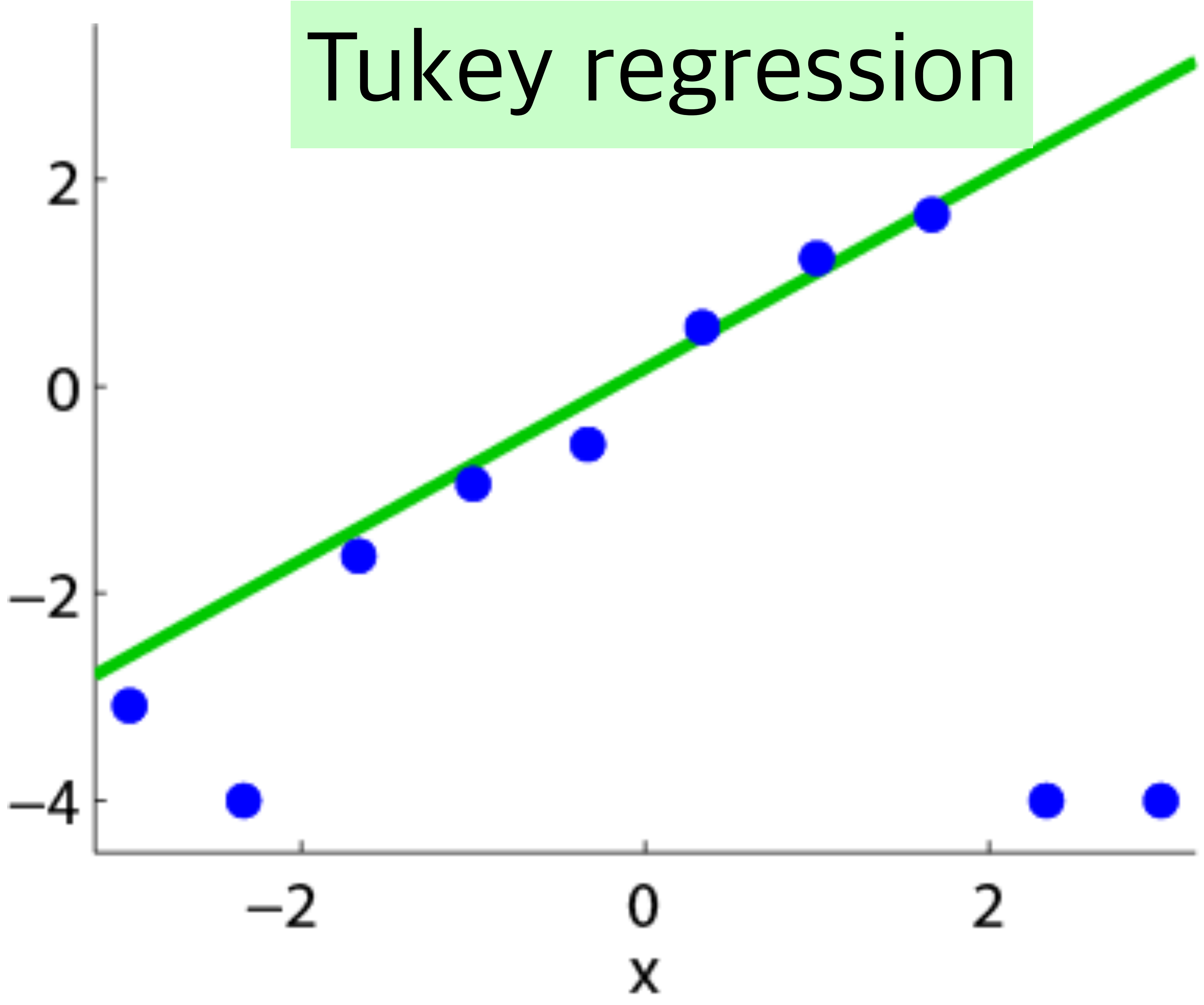
$$w_{\text{Huber}} = \begin{cases} 1 & (|r| \leq \eta) \\ \eta/|r| & (|r| > \eta) \end{cases}$$



Simple example

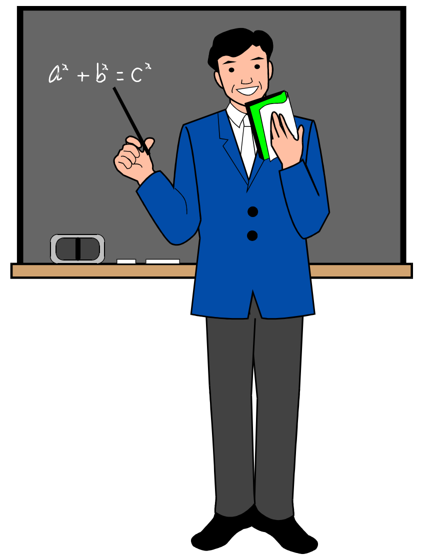
- Tukey regression is more robust to outliers.
- Since it is a non-convex optimization problem, the solution may depend on initialization.





Summary of robust regression

- Squared loss (mean) is vulnerable to outliers
- Absolute loss (median) is robust to outliers
- Huber loss balances robustness and efficiency
  - Solution cannot be obtained analytically
- Tukey loss improves robustness further
  - May need to be a bit more cautious about optimization



Summary of regression

1. Learning models
2. Least squares regression
3. Regularized regression
4. Sparse regression
5. Robust regression

Summary of regression

- Models for learning functions
  - Linear models
  - Kernel models
  - Non-linear models
- Least-squares regression
  - Minimizes the squared error with the training sample
  - Solutions can be calculated analytically
- Online regression
  - Sequential learning by extracting data one at a time
  - Large amounts of data can be handled

Summary of regression

- ℓ_2-regularized regression
  - Reduces over-fitting of least-squares regression
  - Solutions can be calculated analytically
  - Cross-validation is used for model selection
- ℓ_1-regularized regression (sparse regression)
  - Data where many of the true parameter values are zero can be trained properly.
- Robust regression
  - Enhanced robustness against outliers.

Schedule

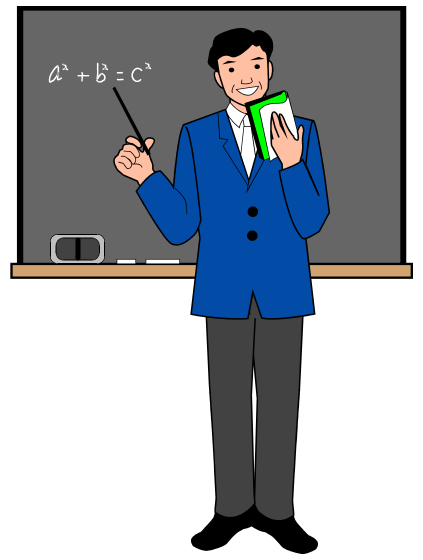
1. 04/8 Introduction
2. 04/15 Regression 1
3. 04/22 Regression 2

- 04/30 Cancelled

4. 05/13 Classification 1
5. 05/20 Classification 2
6. 05/27 Deep learning 1

- 06/03 No lecture

7. 06/10 Deep learning 2
8. 06/17 Deep learning 3
9. 06/24 Semi-supervised learning
10. 07/01 Language models
11. 07/08 Representation learning 1
12. 07/15 Representation learning 2
13. 07/22 Advanced topics



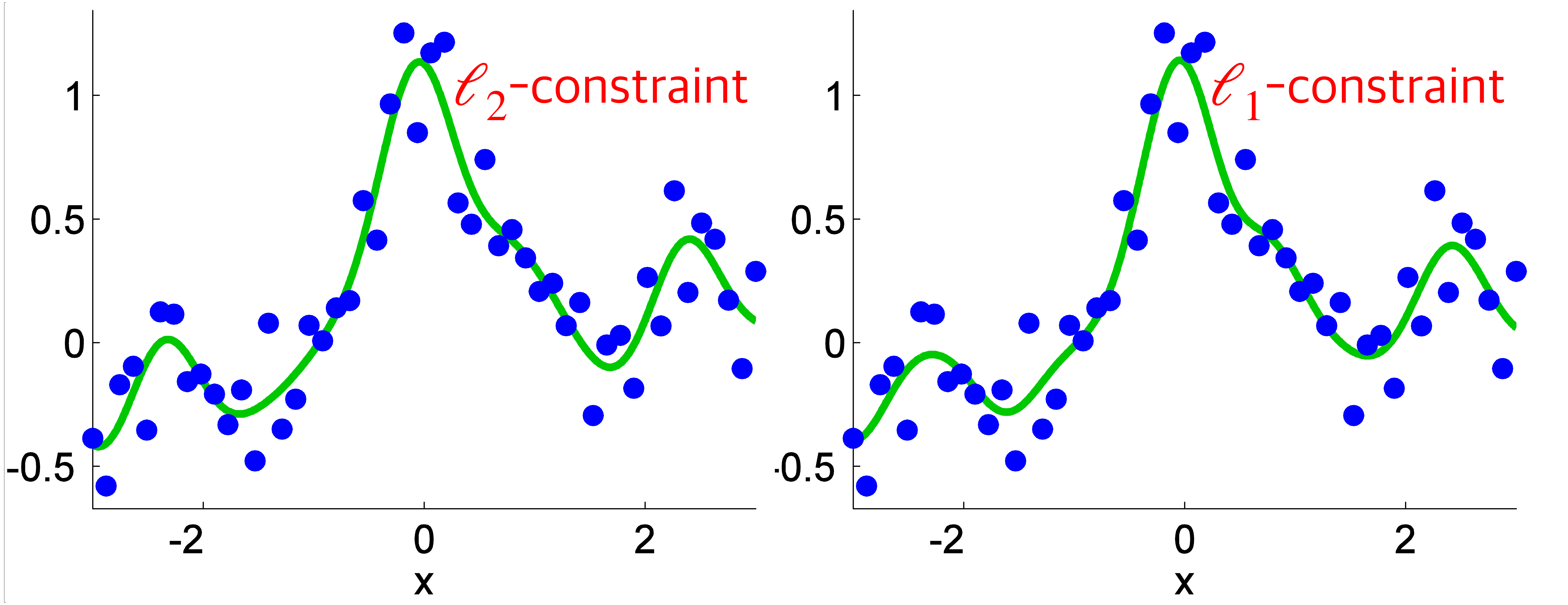
Coming up next

- Classification 1

Homework 1

- Implement the ℓ_1-constraint LS (iteratively reweighted shrinkage). You may use the same data from previous week. (No need to do cross validation).

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



Homework 2

- Setup
  - Linear model: $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=1}^{b} \theta_j \phi_j(\boldsymbol{x})$
  - Basis functions: $\{\phi_j(\boldsymbol{x})\}_{j=1}^{b}$
- Prove that the solution to the weighted LS method is the following: $\widehat{\boldsymbol{\theta}} = (\boldsymbol{\Phi}^\top \widetilde{\boldsymbol{W}} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \widetilde{\boldsymbol{W}} \boldsymbol{y}$
- Weighted LS problem: $\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} \widetilde{w}_i \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i \right)^2$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \cdots & \phi_b(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_n) & \cdots & \phi_b(\boldsymbol{x}_n) \end{pmatrix}$$

$$\widetilde{\boldsymbol{W}} = \mathrm{diag}\left(\widetilde{w}_1, \ldots, \widetilde{w}_n\right)$$

$$\boldsymbol{y} = (y_1, \ldots, y_n)^\top$$