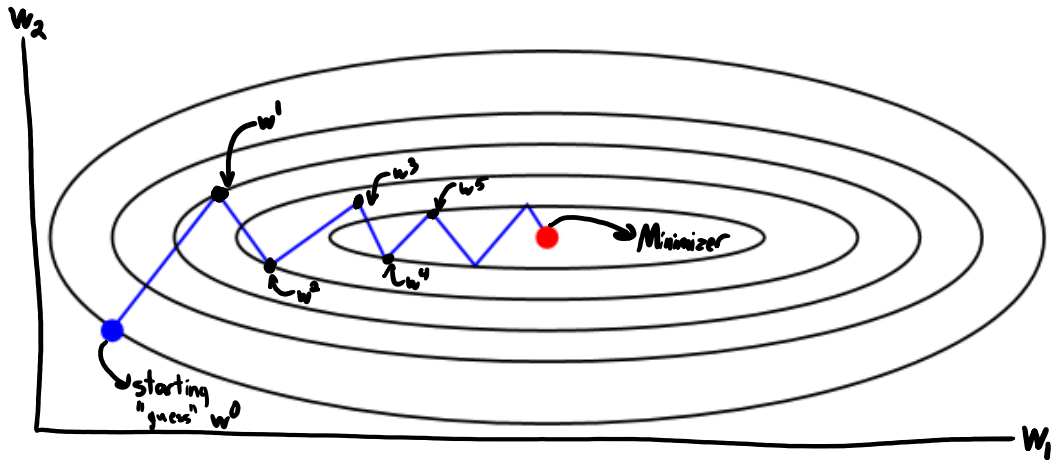
CPSC 340:
Machine Learning and Data Mining

Robust Regression

Last Time: Gradient Descent and Convexity

- We introduced gradient descent:
  - Uses sequence of iterations of the form:

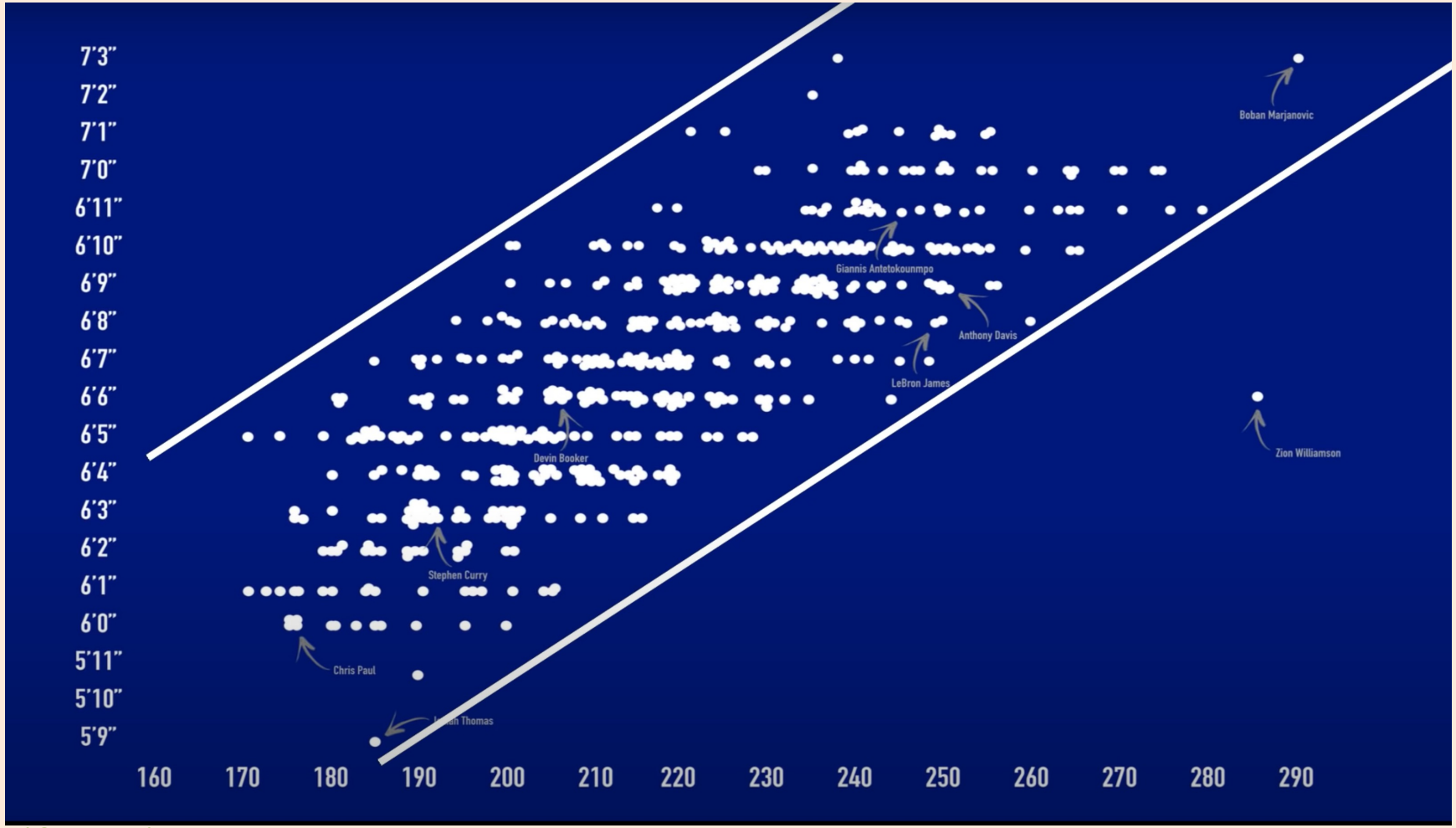
$$w^{t+1} = w^t - \alpha^t \nabla f(w^t)$$



  - Converges to a stationary point where $\nabla f(w) = 0$ under weak conditions.
    - Will be a global minimum if the function is convex.
- We discussed ways to show a function is convex:
  - Second derivative is non-negative (1D functions).
  - Closed under addition, multiplication by non-negative, maximization.
  - Any [squared-]norm is convex.
  - Composition of convex function with linear function is convex.

Least Squares with Outliers

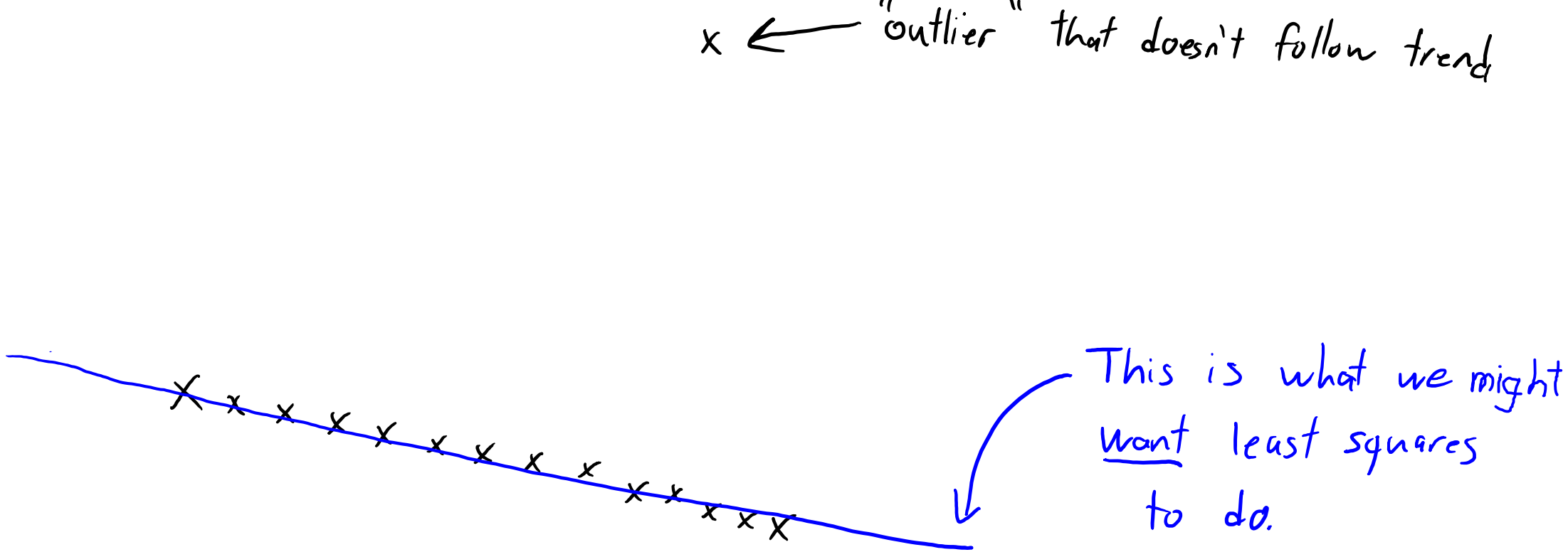
- Height vs. weight of NBA players:



https://www.youtube.com/watch?v=i4eYWl1ewFo

Least Squares with Outliers

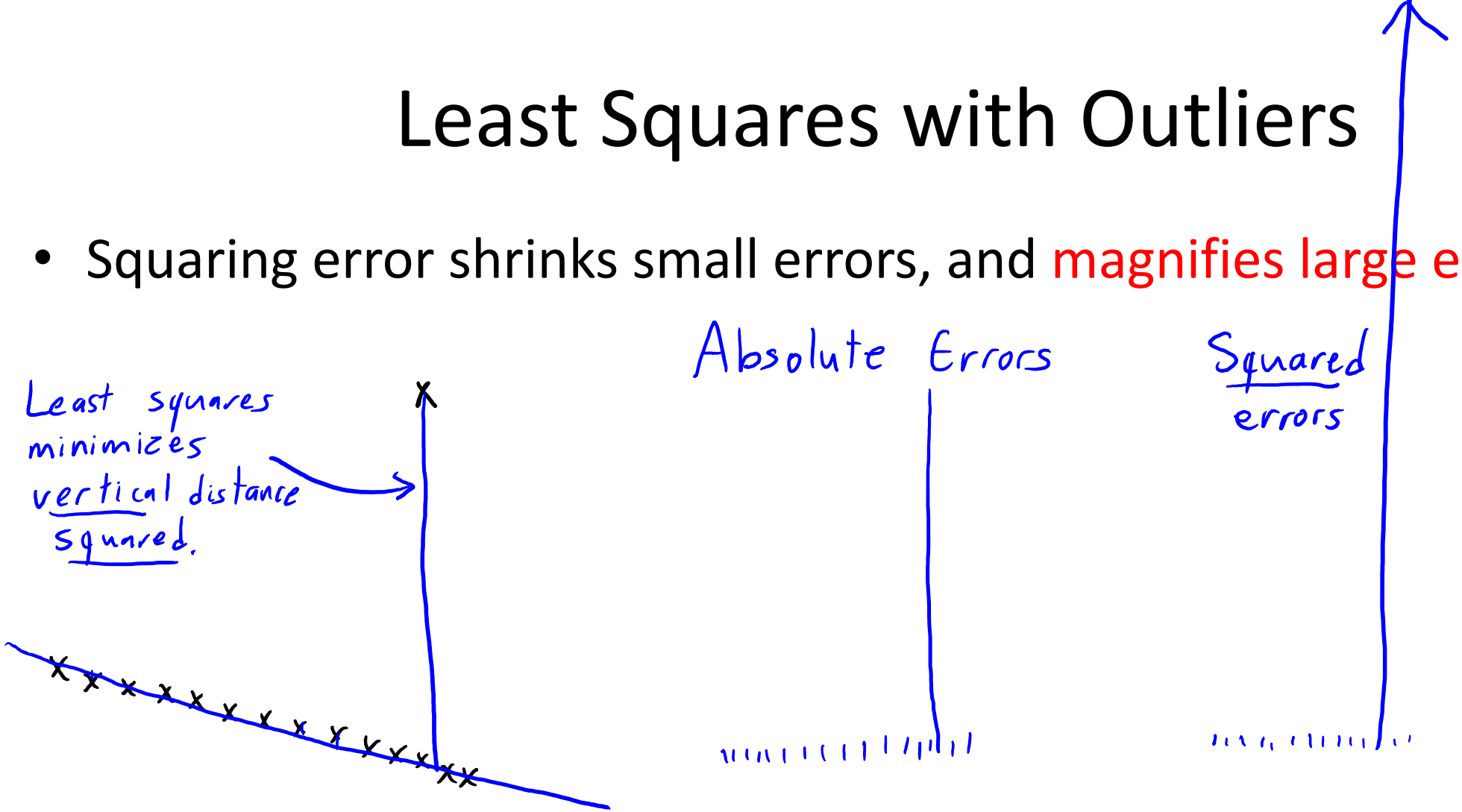
- Consider least squares problem with outliers in 'y':



http://setosa.io/ev/ordinary-least-squares-regression

Least Squares with Outliers

- Squaring error shrinks small errors, and magnifies large errors:

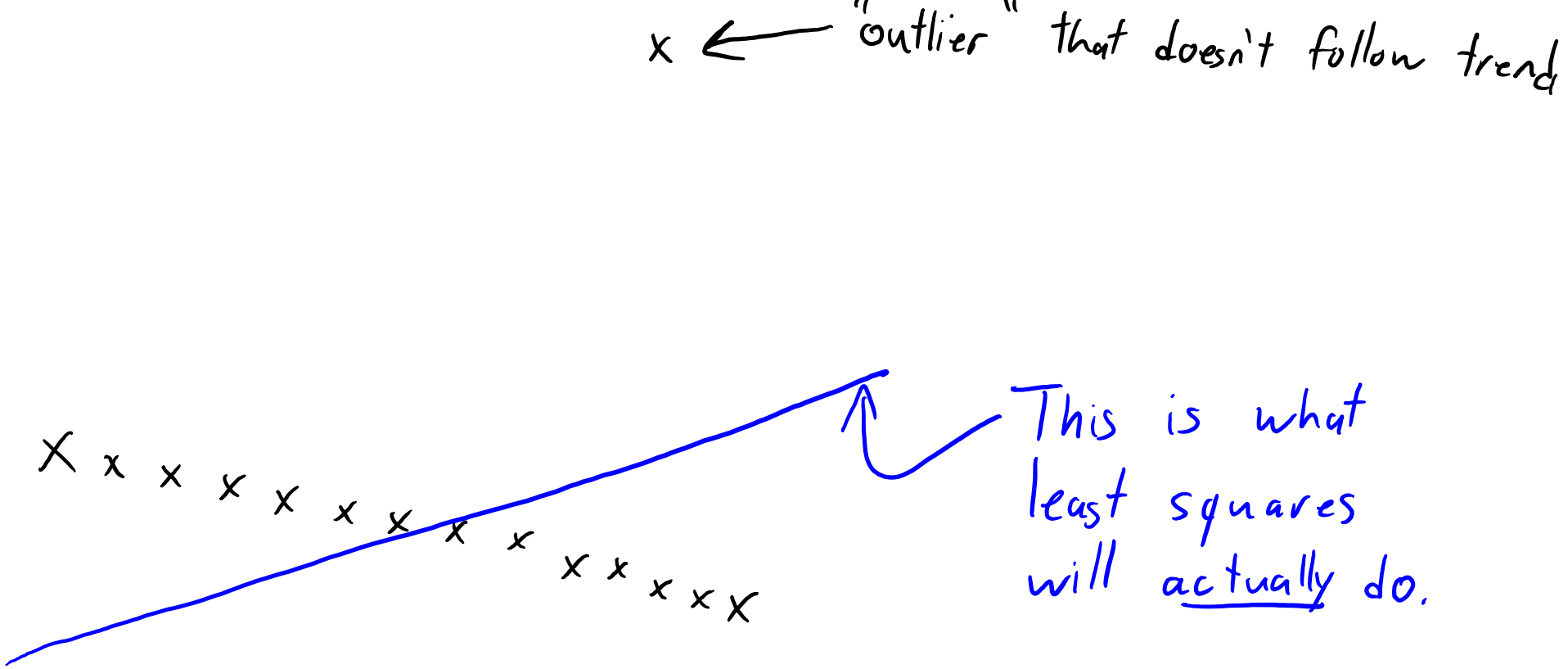


- Outliers (large error) influence ‘w’ much more than other points.

https://seeing-theory.brown.edu/regression-analysis/index.html

Least Squares with Outliers

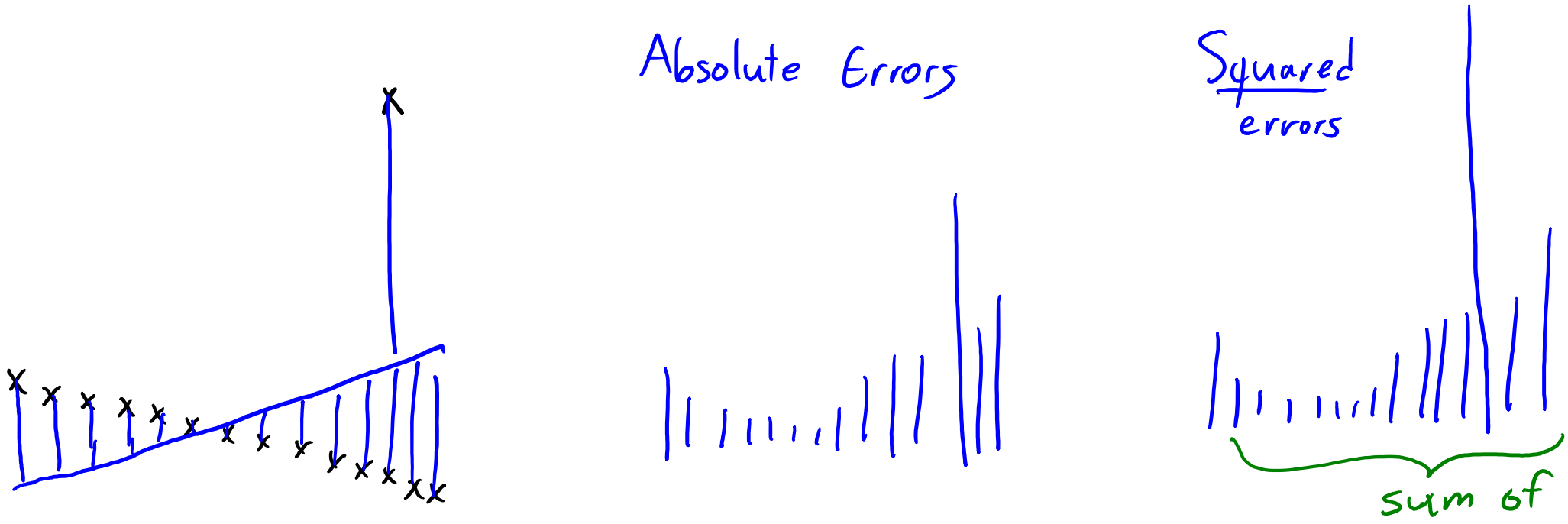
- Consider least squares problem with outliers in 'y':



- Least squares is very sensitive to outliers.

Least Squares with Outliers

- Squaring error shrinks small errors, and magnifies large errors:



- Outliers (large error) influence 'w' much more than other points.
  - Good if outlier means 'plane crashes', bad if it means 'data entry error'.

Robust Regression

- Robust regression objectives focus less on large errors (outliers).
- For example, use absolute error instead of squared error:

$$f(w) = \sum_{i=1}^{n} |w^T x_i - y_i|$$

- Now decreasing 'small' and 'large' errors is equally important.
- Instead of minimizing L2-norm, minimizes L1-norm of residuals (Least Absolute Deviation):

Least squares:

$$f(w) = \frac{1}{2} \|Xw - y\|^2$$

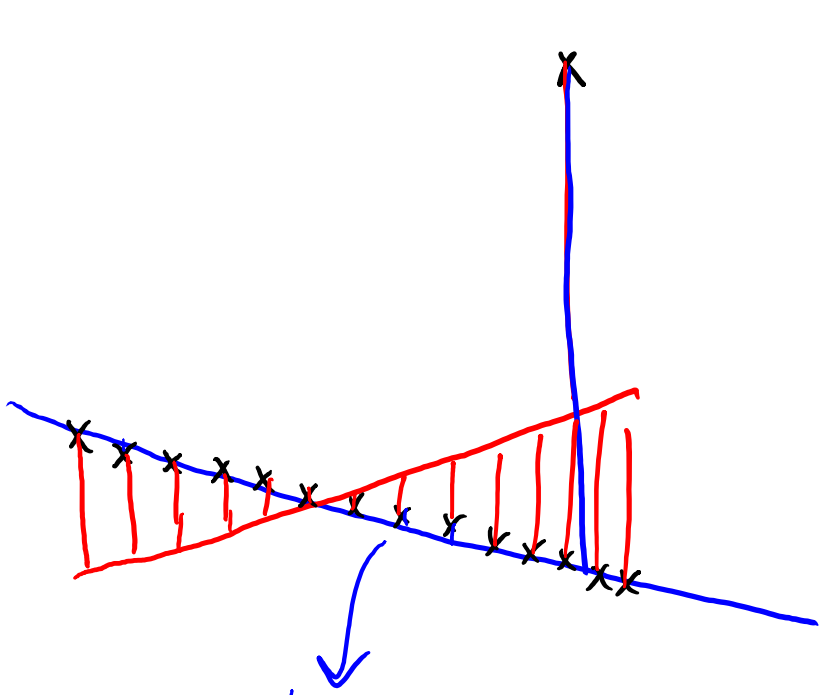
Least absolute error:

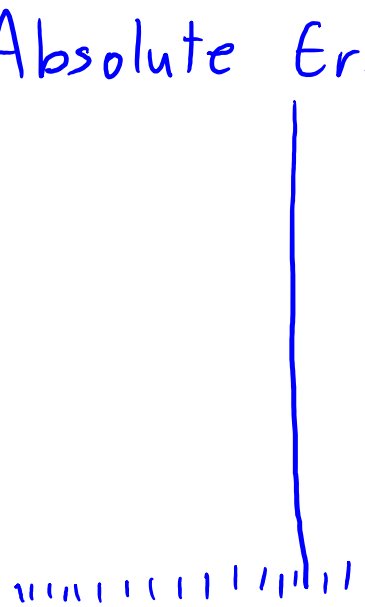
$$f(w) = \|Xw - y\|_1$$

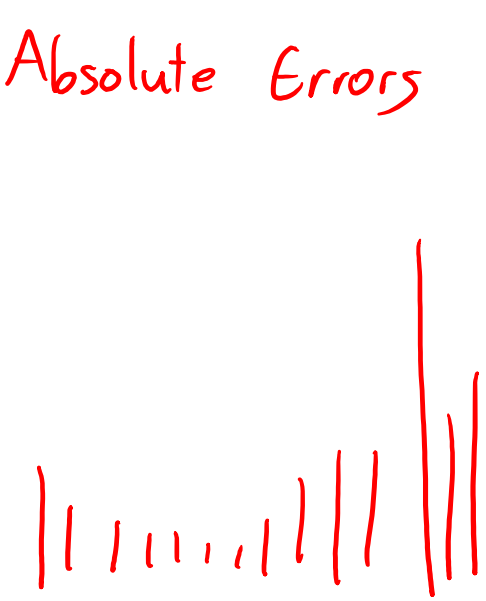
$$\sum_{i=1}^{n} |w^T x_i - y_i| = \sum_{i=1}^{n} |r_i| = \|r\|_1 = \|Xw - y\|_1$$

Least Squares with Outliers

- Absolute error is more robust to outliers:







Linear model 'w' minimizing $f(w) = \|Xw - y\|_1 = \sum_{i=1}^{n} |w^T x_i - y_i|$

Least Square vs Least Absolute Deviation

The least absolute deviation regression was introduced around 50 years before the **least-squares method**, in 1757, by **Roger Joseph Boscovich**. He used this procedure while trying to reconcile incoherent measures that were used to estimate the shape of the earth. **Pierre Simon de Laplace** adopted this method 30 years later, yet it was obscured under the shadow of the least-squares method developed by **Adrien Marie Legendre** and **Carl Friedrich Gauss**. The easy calculus of the least-squares method made least squares much more popular than the LAD method. Yet in recent years and with advances in statistical computing, the LAD method can be easily used.

"The method of least squares, when applied to a system of observations, in which one of the extreme errors is very great, does not generally give so correct a result as the method proposed by Boscovich [...]; the reason is, that in the former method, this extreme error [like any other] affects the result in proportion to the ***second power of the error****; but in the other method, it is as the first power."* — Bowditch (c.1830) (source)

https://link.springer.com/referenceworkentry/10.1007/978-0-387-32833-1_225

https://www.cantorsparadise.com/least-squares-vs-least-absolute-errors-a-250-year-old-debate-bf102929a80f

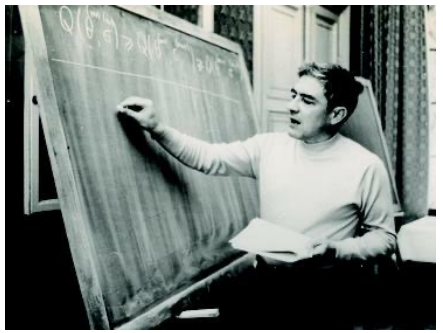
Least Square vs Least Absolute Deviation

Similar to Gauss and Legendre, Eddington was also working on estimation problems in astronomy. In his 1914 work *Stellar Movements and The Structure of The Universe,* he states that

"in calculating the mean error of a series of observations it is preferable to use the simple mean residual irrespective of sign rather than the mean square residual ... ... this is contrary to the advice of most textbooks, but it can be shown to be true'"
— Eddington (1914) (source)

"thus it becomes painfully clear that the naturally occurring deviations from the idealized model are large enough to render meaningless the traditional asymptotic optimality theory". — Peter Huber (1981)

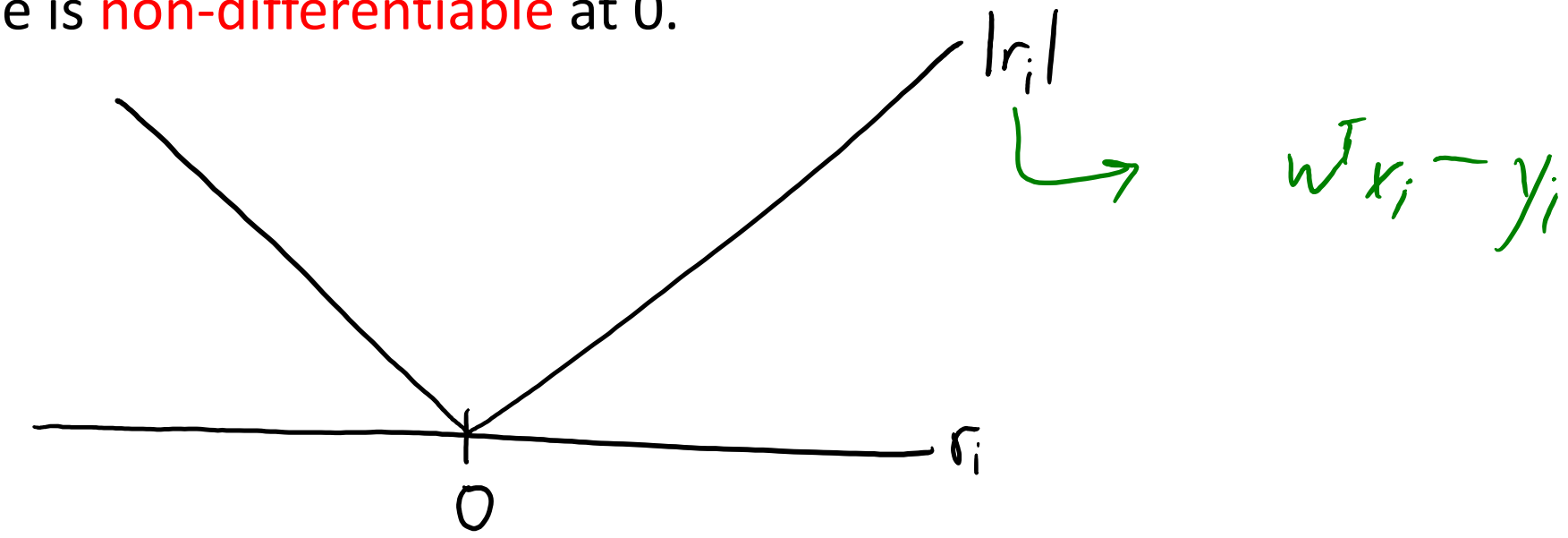
Sir Arthur Eddington



Peter Huber

Regression with the L1-Norm

- Unfortunately, minimizing the absolute error is harder.
  - We don't have "normal equations" for minimizing the L1-norm.
  - Absolute value is non-differentiable at 0.



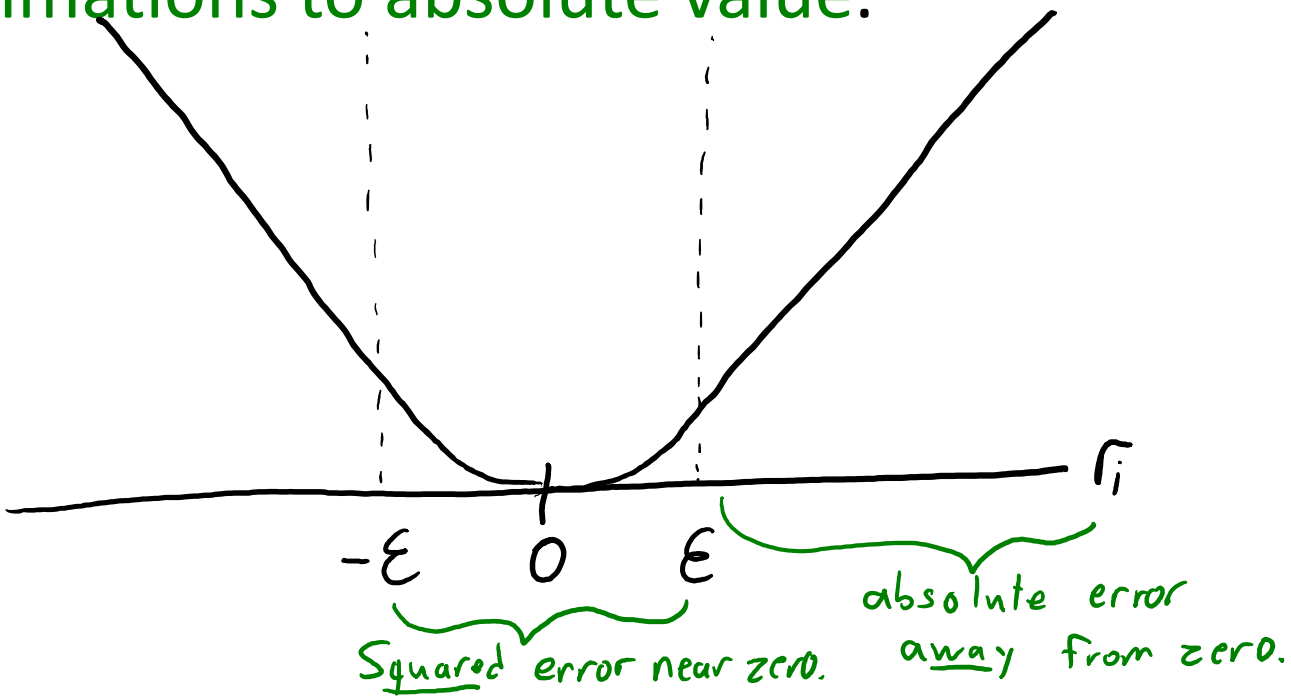
  - Generally, harder to minimize non-smooth than smooth functions.
    - Unlike smooth functions, the gradient may not get smaller near a minimizer.
  - To apply gradient descent, we'll use a smooth approximation.

Smooth Approximations to the L1-Norm

- There are differentiable approximations to absolute value.
  - Common example is Huber loss:

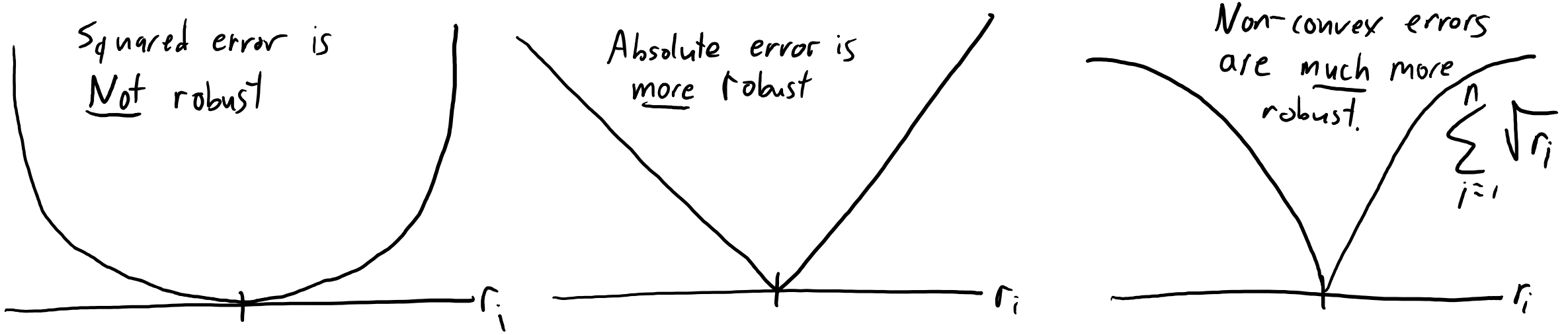
$$f(w) = \sum_{i=1}^{n} h(w^T x_i - y_i)$$

$$h(r_i) = \begin{cases} \frac{1}{2} r_i^2 & \text{for } |r_i| \leq \varepsilon \\ \varepsilon(|r_i| - \frac{1}{2}\varepsilon) & \text{otherwise} \end{cases}$$

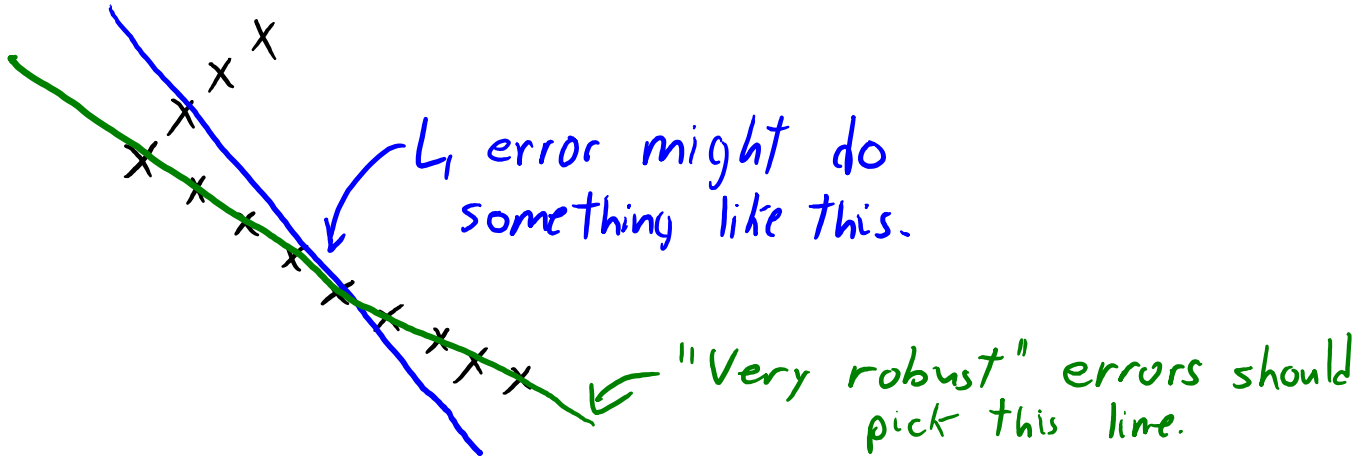


  - Note that 'h' is differentiable: $h'(\varepsilon) = \varepsilon$ and $h'(-\varepsilon) = -\varepsilon$.
  - This 'f' is convex but setting $\nabla f(x) = 0$ does not give a linear system.
    - But we can minimize the Huber loss using gradient descent.

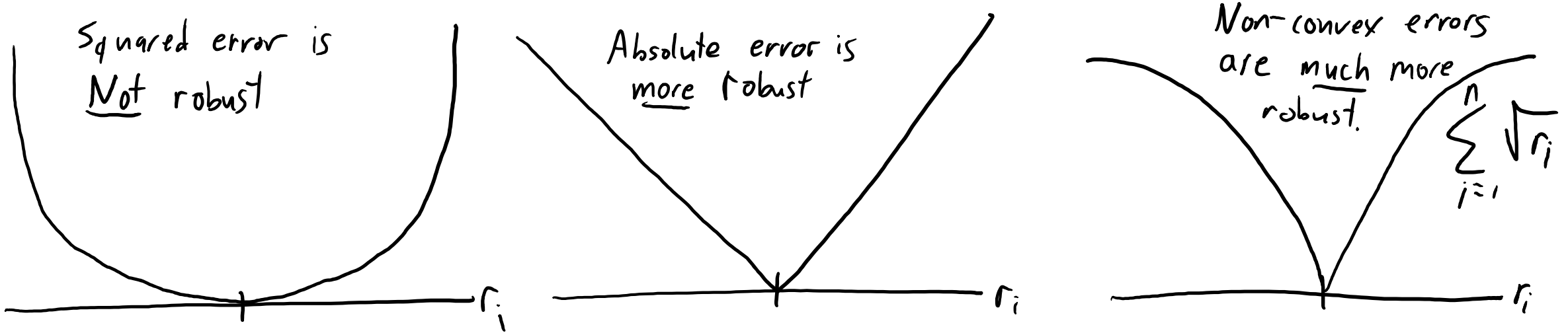
Very Robust Regression



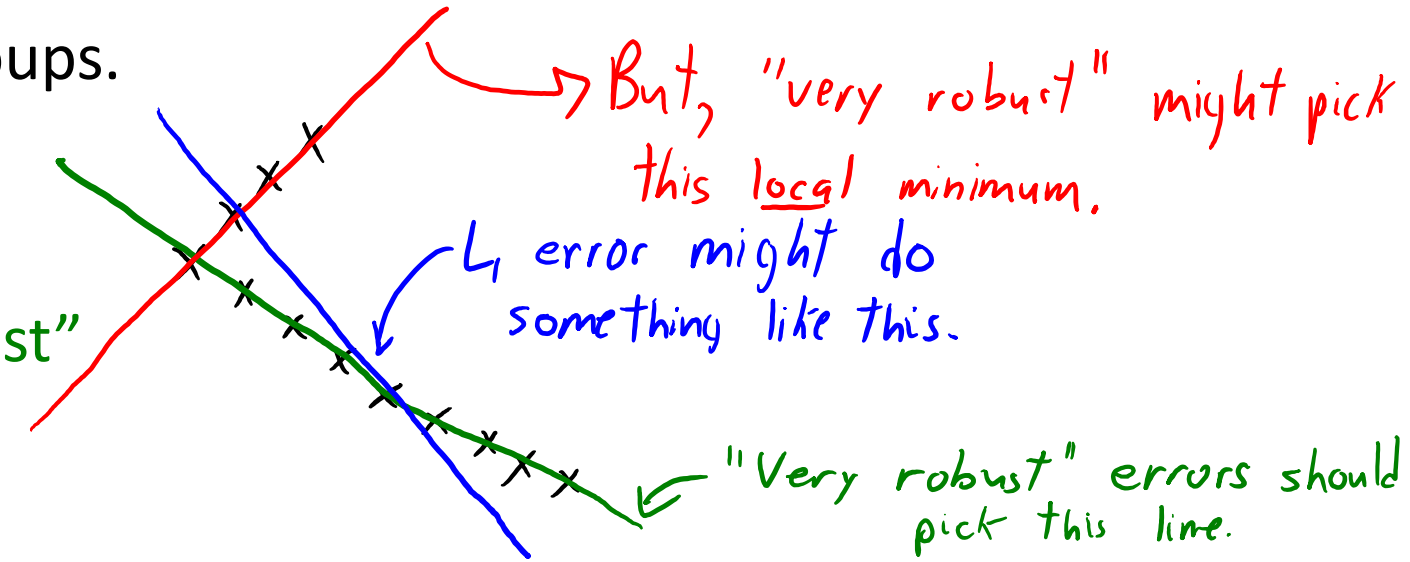
- Non-convex errors can be very robust:
  - Not influenced by outlier groups.



Very Robust Regression

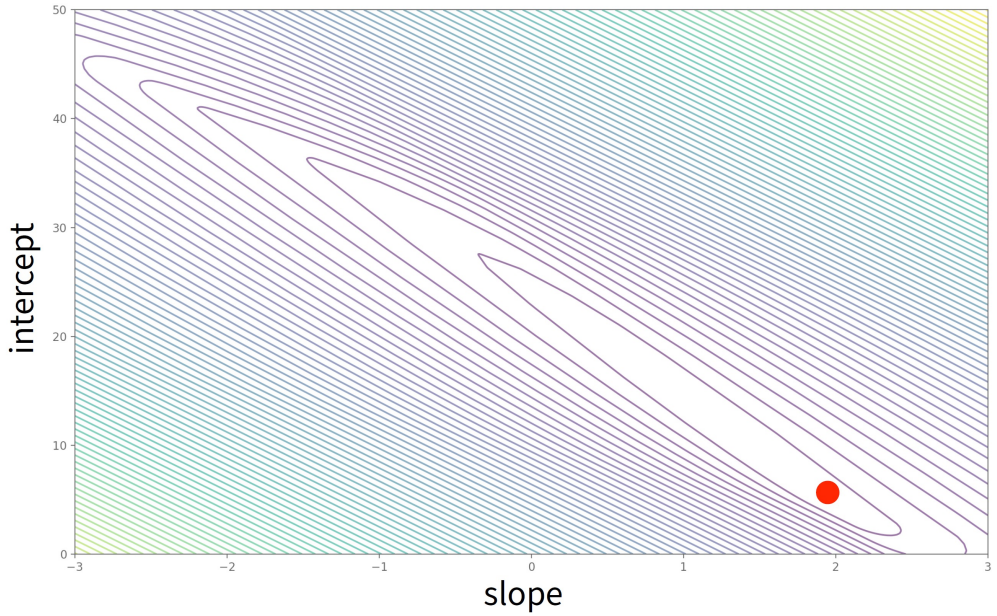


- Non-convex errors can be very robust:
  - Not influenced by outlier groups.
  - But non-convex, so finding global minimum is hard.
  - Absolute value is "most robust" convex loss function.

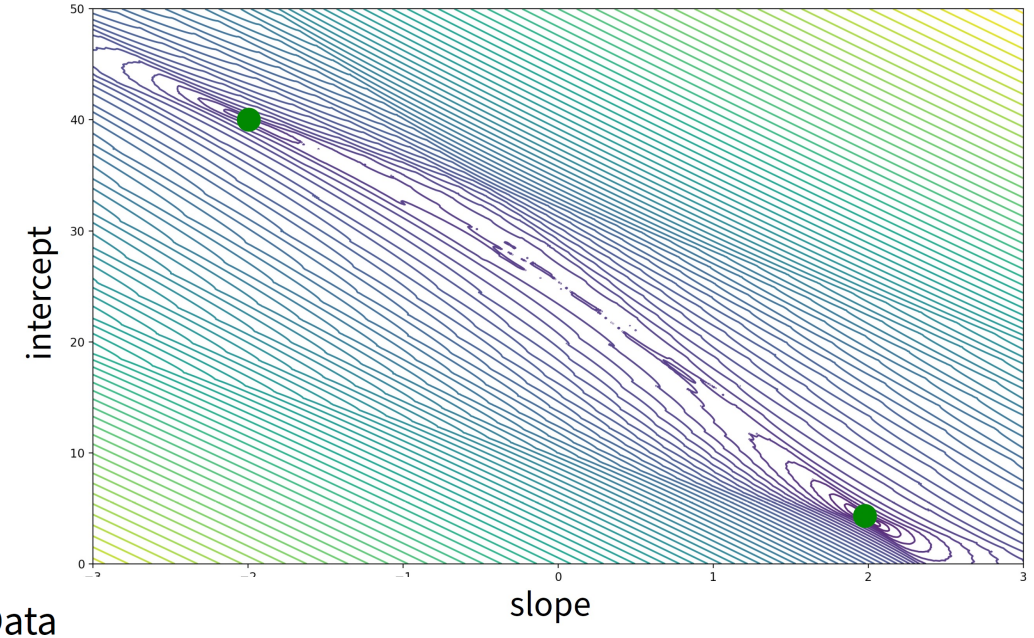


Very Robust Regression

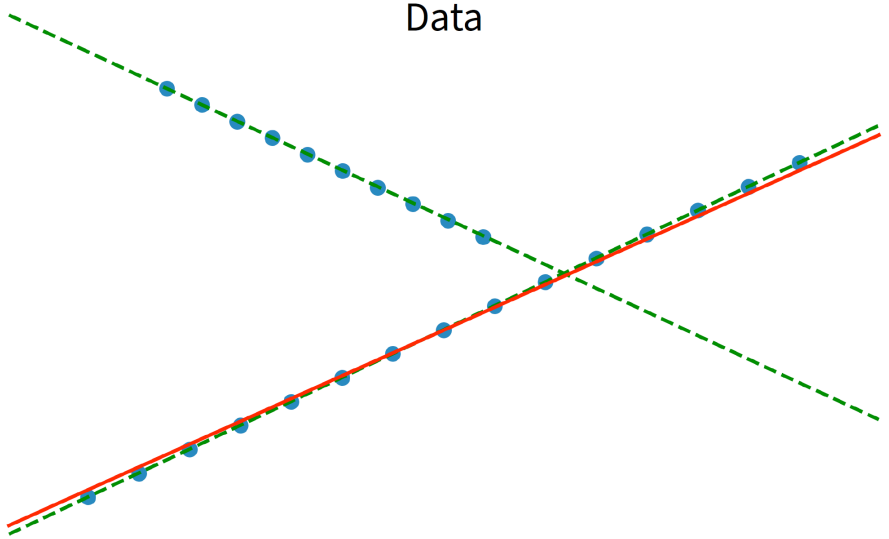
Absolute Error (convex)



Square Root Error (non-convex)



Data

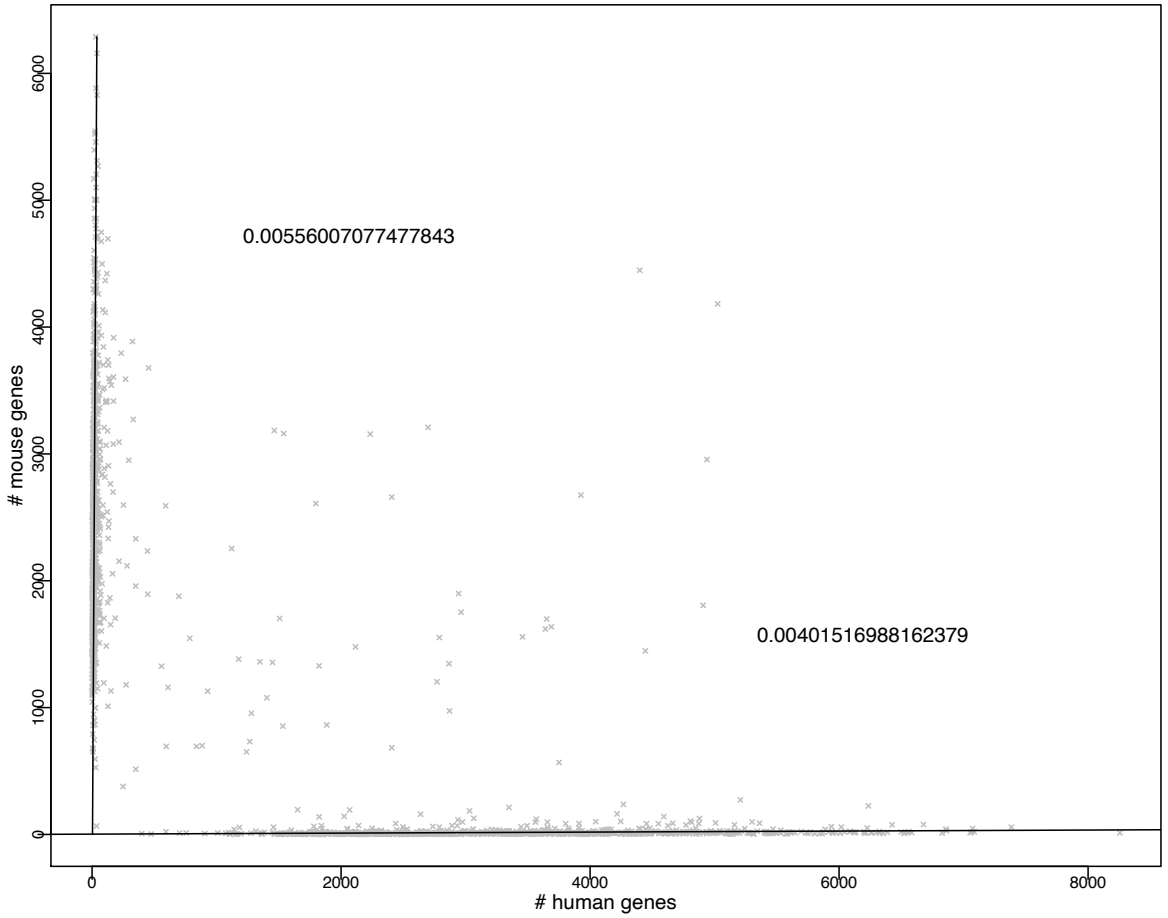


data points

local minimum (square root error)

global minimum (absolute error)

Is Robust Regression Used?

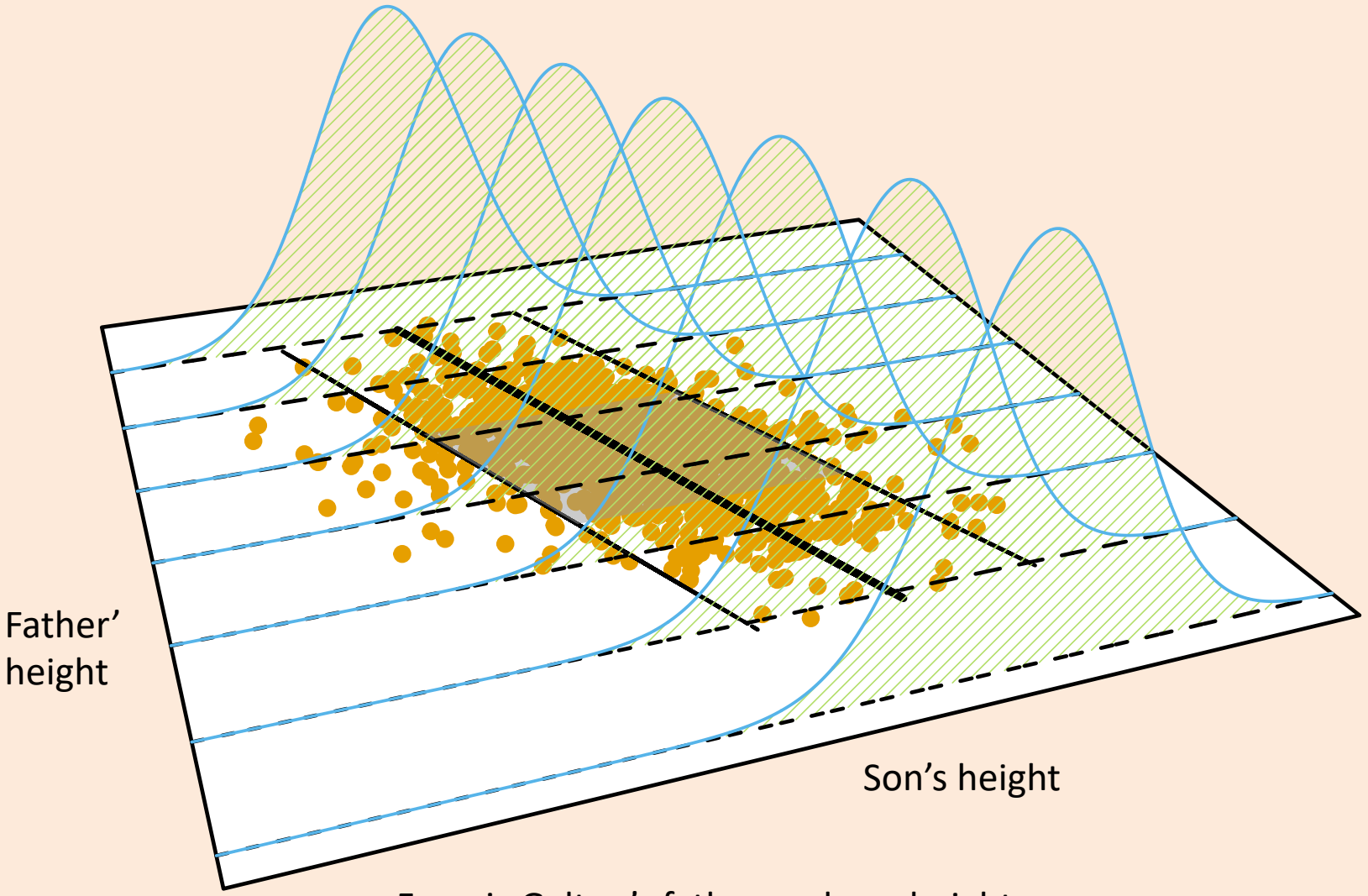


- Because real data are typically noisy, and robust regression generally produce better results

- Not widely used because, e.g., 1) software, 2) we already have least square, 3) people don't familiar with robust regression, …

https://www.nature.com/articles/s41587-020-0465-8/figures/12

Least Square and the Gaussian Distribution



Francis Galton's father and son heights

$r_i = w^T x_i - y_i$ follows a normal distribution $N(0, \sigma^2)$

For LAD, the residual follows a Laplace distribution

Next Topic: Brittle Regression

Motivation for Modeling Outliers

- What if the “outlier” is the only male person in your dataset?
  - Do you want to be robust to the outlier?
  - Will the model work for everyone if it has good average case performance?

https://xkcd.com/937/

Accuracy vs. Fairness Trade-Off?

- Improving test error might need to make fairness worse?
  - If you chase you outliers you might worsen generalization.
  - If you do not chase the outliers you might worsen fairness.
- Recent related paper:

Inherent Tradeoffs in Learning Fair Representations

Han Zhao, Geoffrey J. Gordon; 23(57):1–26, 2022.

Abstract

Real-world applications of machine learning tools in high-stakes domains are often regulated to be fair, in the sense that the predicted target should satisfy some quantitative notion of parity with respect to a protected attribute. However, the exact tradeoff between fairness and accuracy is not entirely clear, even for the basic paradigm of classification problems. In this paper, we characterize an inherent tradeoff between statistical parity and accuracy in the classification setting by providing a lower bound on the sum of group-wise errors of any fair classifiers. Our impossibility theorem could be interpreted as a certain uncertainty principle in fairness: if the base rates differ among groups, then any fair classifier satisfying statistical parity has to incur a large error on at least one of the groups. We further extend this result to give a lower bound on the joint error of any (approximately) fair classifiers, from the perspective of learning fair representations. To show that our lower bound is tight, assuming oracle access to Bayes (potentially unfair) classifiers, we also construct an algorithm that returns a randomized classifier which is both optimal (in terms of accuracy) and fair. Interestingly, when the protected attribute can take more than two values, an extension of this lower bound does not admit an analytic solution. Nevertheless, in this case, we show that the lower bound can be efficiently computed by solving a linear program, which we term as the TV-Barycenter problem, a barycenter problem under the TV-distance. On the upside, we prove that if the group-wise Bayes optimal classifiers are close, then learning fair representations leads to an alternative notion of fairness, known as the accuracy parity, which states that the error rates are close between groups. Finally, we also conduct experiments on real-world datasets to confirm our theoretical findings.

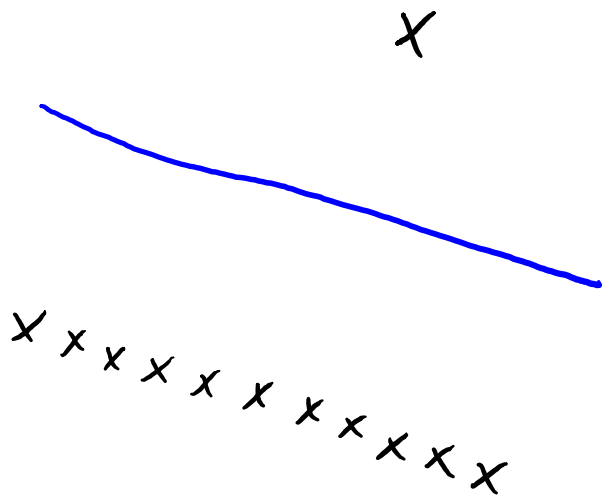
https://www.jmlr.org/papers/v23/21-1427.html

“Brittle” Regression

- What if you really care about getting the outliers right?
  - You want to minimize size of worst error across examples.
    - For example, if in worst case the plane can crash or you perform badly on a group.
- In this case you could use something like the infinity-norm:

$$f(w) = \|Xw - y\|_\infty$$

where $\|r\|_\infty = \max_i \{|r_i|\}$

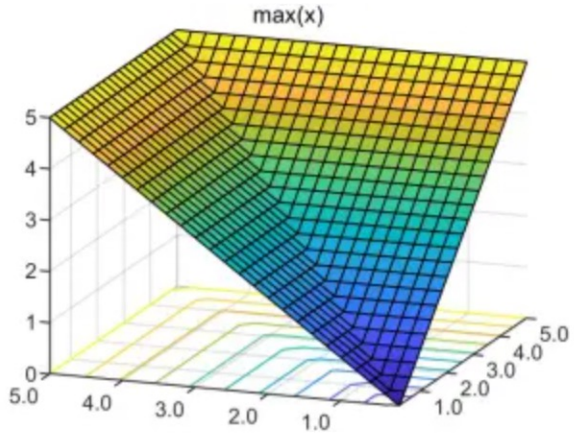


- Very sensitive to outliers (“brittle”), but minimizes highest error.
  - Different than previous errors, which were all some sort of average.

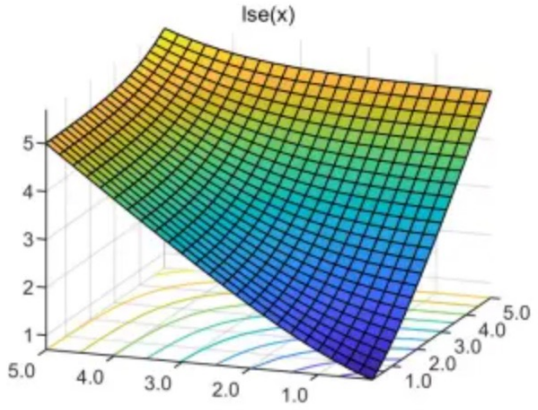
Log-Sum-Exp Function

- As with the L_1-norm, the L_∞-norm is convex but non-smooth:
  - We can again use a smooth approximation and fit it with gradient descent.

- Convex and smooth approximation to max function is log-sum-exp function:



$$\max_i \{z_i\} \approx \log\left(\sum_i \exp(z_i)\right)$$



  - We will use this several times in the course.
  - Notation alert: when I write "log" I always mean "natural" logarithm: log(e) = 1.

- Intuition behind log-sum-exp:
  - $\sum_i \exp(z_i) \approx \max_i \exp(z_i)$, as largest element is magnified exponentially (if no ties).
  - And notice that $\log(\exp(z_i)) = z_i$.

Log-Sum-Exp Function Examples

- Log-sum-exp function as smooth approximation to max:

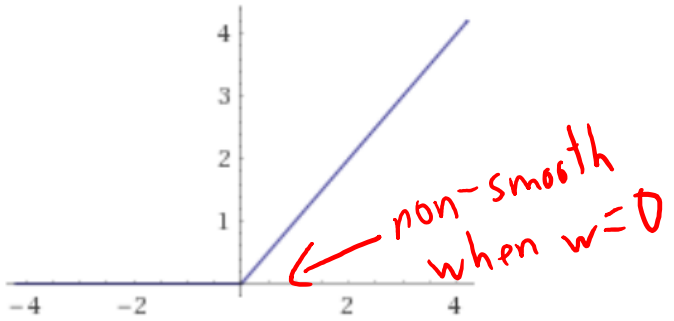
$$\max_i \{z_i\} \approx \log\left(\sum_i \exp(z_i)\right)$$

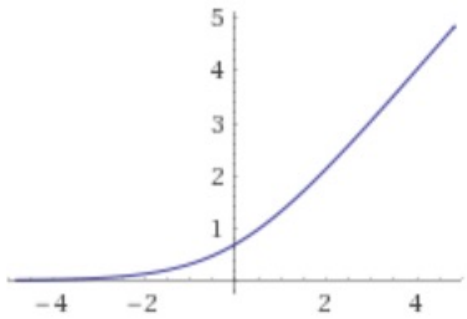
- If there aren't "close" max values, it's really close to the max.

If $z_i = \{2, 20, 5, -100, 7\}$ then $\max_i \{z_i\} = 20$ and $\log\left(\sum_i \exp(z_i)\right) \approx 20.000002$

If $z_i = \{2, 20, 19.9, -100, 7\}$ then $\max_i \{z_i\} = 20$ and $\log\left(\sum_i \exp(z_i)\right) \approx 20.688160$

- Comparison of max{0,w} and smooth log(exp(0) + exp(w)):





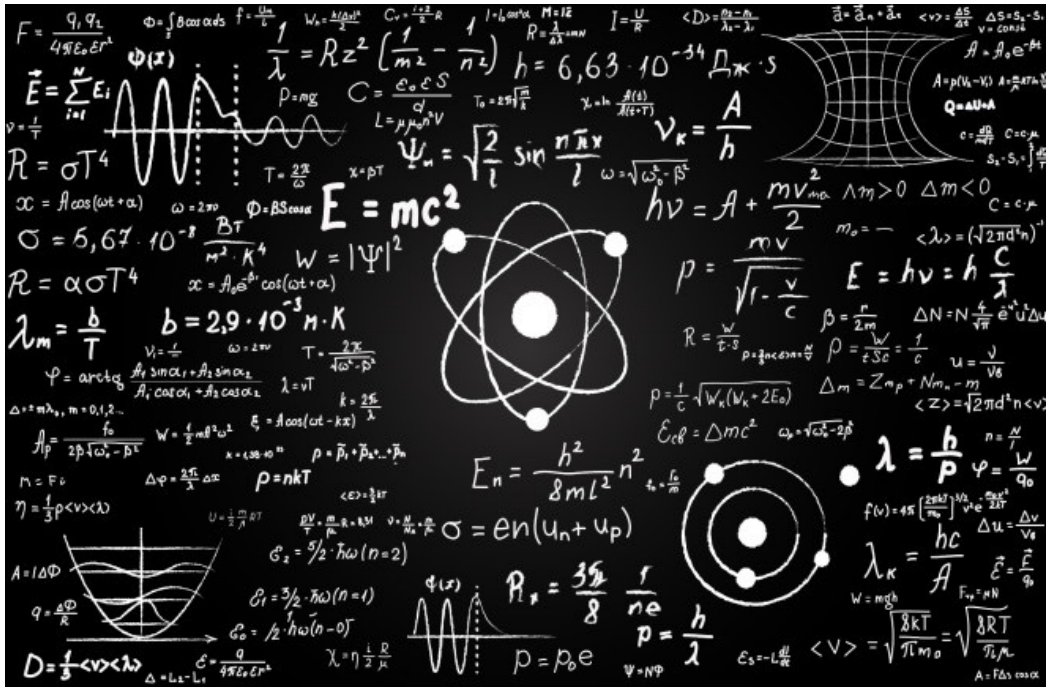
Next Topic: Model Selection

Finding the “True” Model

- To measure performance we have focused on minimize test error.
  - This is good if our goal is to predict on well on new IID data
  - But this is a weird way to do science!
    - “It's true there's been a lot of work on trying to apply statistical models to various linguistic problems. I think there have been some successes, but a lot of failures. There is a notion of success ... which I think is novel in the history of science. It interprets success as approximating unanalyzed data.” Noam Chomsky.

Finding the “True” Model

- To measure performance we have focused on minimize test error.
  - This is good if our goal is to predict on well on new IID data
  - But this is a weird way to do science!
- We normally want to design simple models that explain the world.
  - Might work even if new data is not IID!

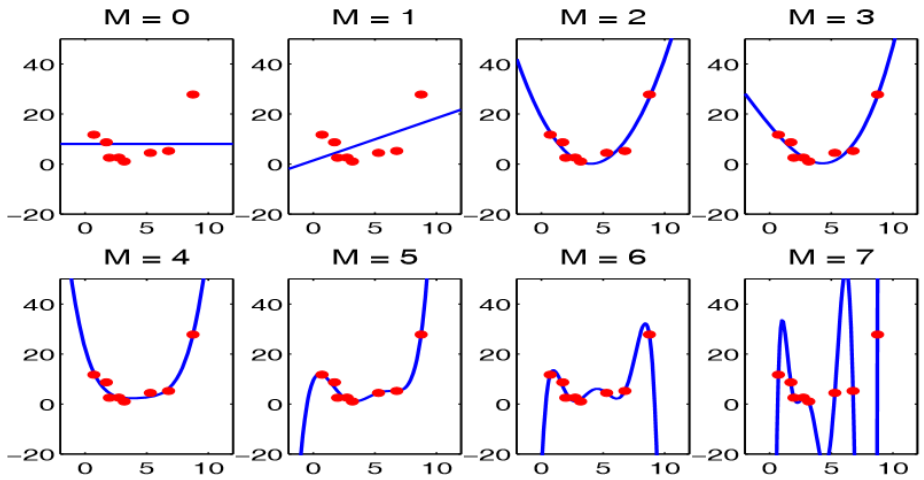


  - Next topic: finding the “true” model.
    - Hint: it is hard unless things are simple!

https://www.discovermagazine.com/the-sciences/the-5-most-important-scientific-equations-of-all-time

Finding the "True" Model

- Consider a simple case of trying to find the "true" model?
  - We believe that y_i really is a polynomial function of x_i.
  - We want to find the degree of the polynomial 'p'.



- Should we choose the 'p' with the lowest training error?
  - No, this will pick a 'p' that is too large.
    (training error always decreases as you increase 'p')

Finding the "True" Model

- Consider a simple case of trying to find the "true" model?
  - We believe that y_i really is a polynomial function of x_i.
  - We want to find the degree of the polynomial 'p'.

- Should we choose the 'p' with the lowest validation error?
  - This will also often choose a 'p' that is too large (due to optimization bias).

  - Consider a case where the true model has p=2 (quadratic function).
    - We fit a quadratic function $y_i = 2x_i^2 - 5$, and a cubic function $y_i = 0.00001x_i^3 + 2x_i^2 - 5$.
    - Cubic is wrong, but by chance might get a lower error on a particular validation set.
      - If we try many models, there are more "chances" to randomly get a lower validation error.

Complexity Penalties

- There are a lot of "scores" people use to find the "true" model.
- Basic idea behind them: put a penalty on the model complexity.
  - Want to **fit the data and have a simple model**.
- For example, minimize training error plus the degree of polynomial.

Let $Z_p = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^p \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^p \\ 1 & x_3 & (x_3)^2 & \cdots & (x_3)^p \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & (x_n)^2 & \cdots & (x_n)^p \end{bmatrix}$

Find 'p' that minimizes:

$$\text{score}(p) = \frac{1}{2}\|Z_p v - y\|^2 + p$$

train error for best 'v' with this basis.

degree of polynomial

  - If we use p=4, use "training error plus 4" as error.
- If two 'p' values have similar error, this prefers the smaller 'p'.

Choosing Degree of Polynomial Basis

- How can we optimize this score?

$$score(p) = \frac{1}{2}||Z_p v - y||^2 + p$$

  - Form Z_0, solve for 'v', compute score(0) = $\frac{1}{2}||Z_0 v - y||^2 + 0$.
  - Form Z_1, solve for 'v', compute score(1) = $\frac{1}{2}||Z_1 v - y||^2 + 1$.
  - Form Z_2, solve for 'v', compute score(2) = $\frac{1}{2}||Z_2 v - y||^2 + 2$.
  - Form Z_3, solve for 'v', compute score(3) = $\frac{1}{2}||Z_3 v - y||^2 + 3$.

  - Choose the degree with the lowest score.
    - "You need to decrease training error by at least 1 to increase degree by 1."

Information Criteria

- There are many scores, usually with the form:

$$\text{score}(p) = \frac{1}{2}\|Z_p v - y\|^2 + \lambda k$$

  - The value 'k' is the "number of estimated parameters" ("degrees of freedom").
    - For polynomial basis, we have k = (p+1).
  - The parameter $\lambda > 0$ controls how strong we penalize complexity.
    - "You need to decrease the training error by least λ to increase 'k' by 1".

- Using ($\lambda = 1$) is called Akaike information criterion (AIC), the RSS is also weighted (the negative loglikelihood).
- Other choices of λ (not necessarily integer) give other criteria:
  - Mallow's C_p.
  - Adjusted R^2.
  - ANOVA-based model selection.

Choosing Degree of Polynomial Basis

- How can we optimize this score in terms of 'p'?

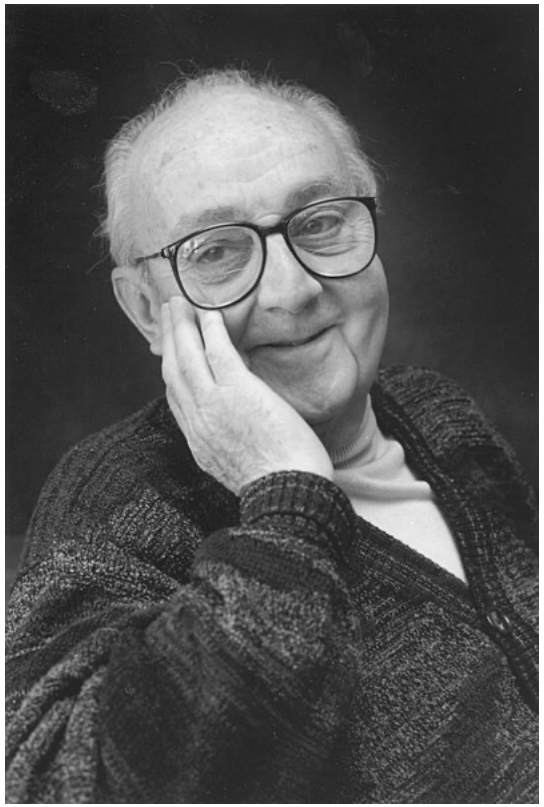
$$\text{score}(p) = \frac{1}{2}\|Z_p v - y\|^2 + \lambda k$$

  - Form Z_0, solve for 'v', compute score(0) = $\frac{1}{2}||Z_0 v - y||^2 + \lambda$.
  - Form Z_1, solve for 'v', compute score(1) = $\frac{1}{2}||Z_1 v - y||^2 + 2\lambda$.
  - Form Z_2, solve for 'v', compute score(2) = $\frac{1}{2}||Z_2 v - y||^2 + 3\lambda$.
  - Form Z_3, solve for 'v', compute score(3) = $\frac{1}{2}||Z_3 v - y||^2 + 4\lambda$.

  - So we need to improve by "at least λ" to justify increasing degree.
    - If λ is big, we'll choose a small degree. If λ is small, we'll choose a large degree.

Why Model Selection?

Since all models are wrong the scientist cannot obtain a "correct" one by excessive elaboration. On the contrary following William of Occam he should seek an economical description of natural phenomena. Just as the ability to devise simple but evocative models is the signature of the great scientist so overelaboration and overparameterization is often the mark of mediocrity.



George E. P. Box

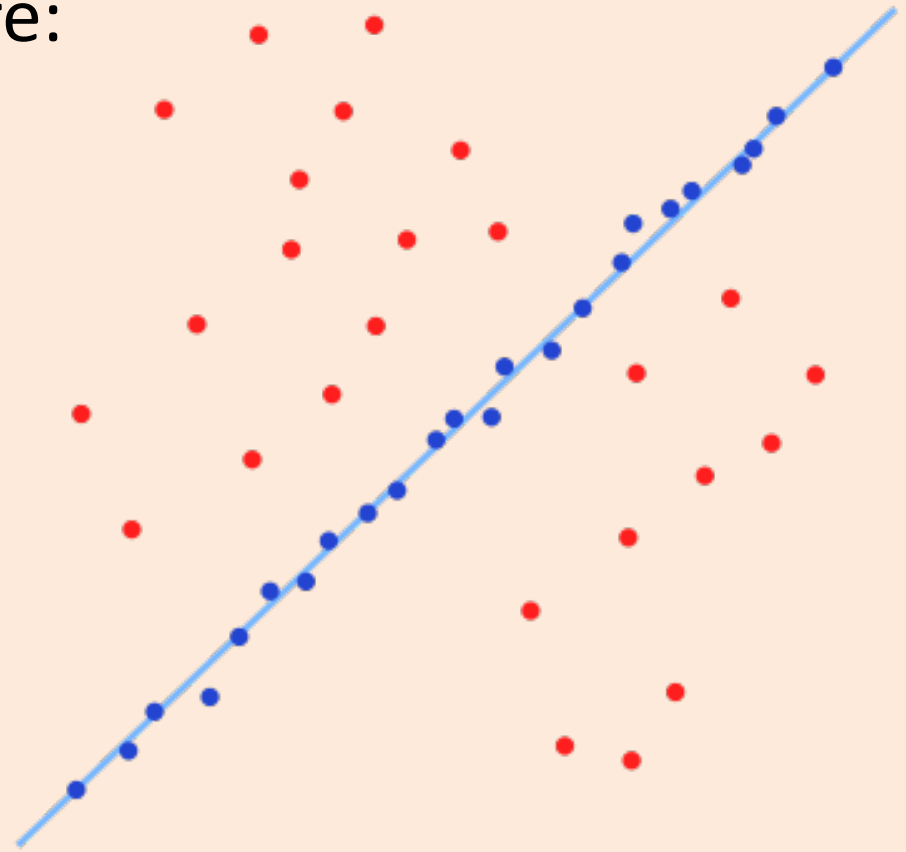
Summary

- Outliers in ‘y’ can cause problem for least squares.
- Robust regression using L1-norm is less sensitive to outliers.
- Brittle regression using Linf-norm is more sensitive to outliers.
- Smooth approximations:
  - Let us apply gradient descent to non-smooth functions.
  - Huber loss is a smooth approximation to absolute value.
  - Log-Sum-Exp is a smooth approximation to maximum.
- Information criteria are scores that penalize number of parameters.
  - When we want to find the “true” model.

- Next time:
  - Can we find the “true” features?

Random Sample Consensus (RANSAC)

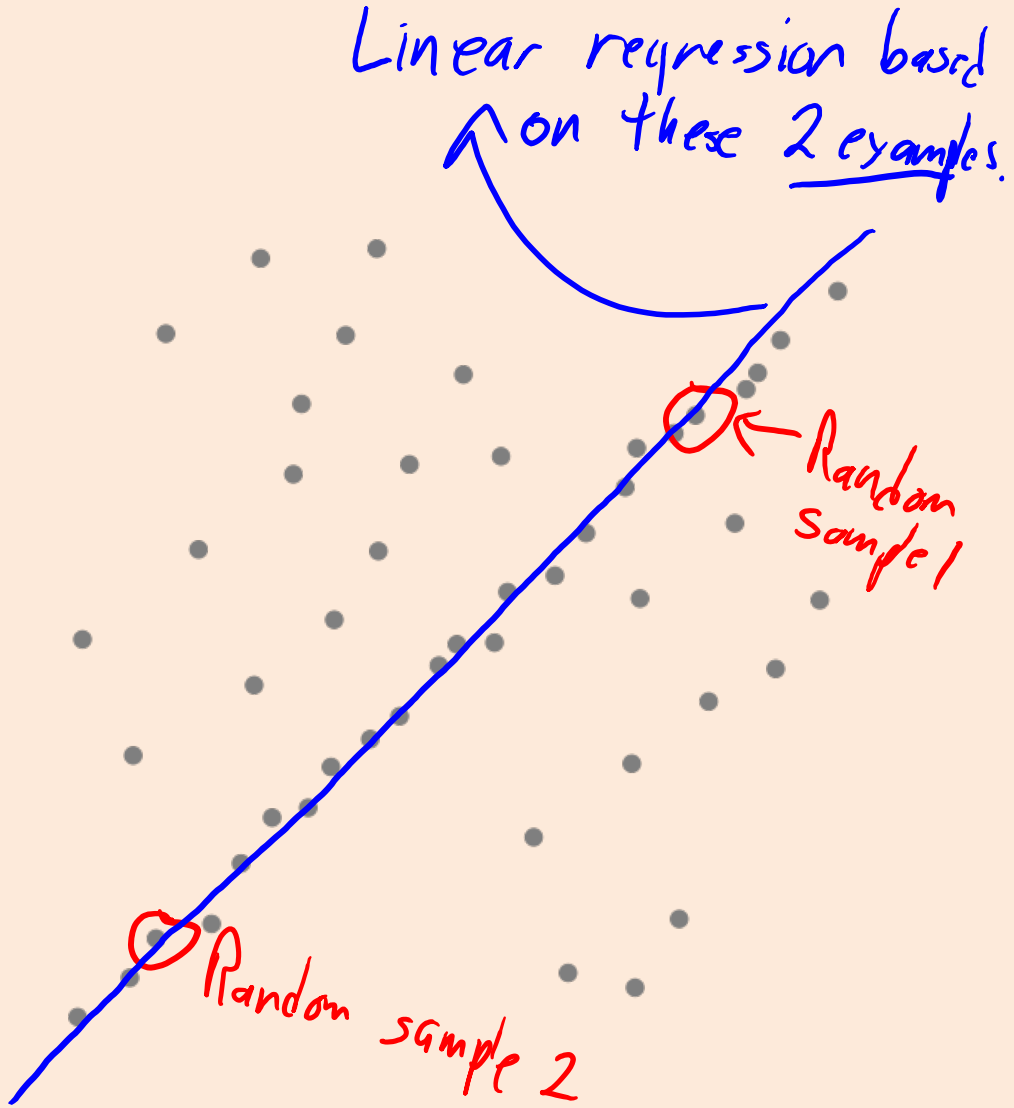
- In computer vision, a widely-used generic framework for robust fitting is random sample consensus (RANSAC).
- This is designed for the scenario where:
  - You have a large number of outliers.
  - Majority of points are “inliers”:
    it’s really easy to get low error on them.



https://en.wikipedia.org/wiki/Random_sample_consensus

Random Sample Consensus (RANSAC)

- RANSAC:
  - Sample a small number of training examples.
    - Minimum number needed to fit the model.
    - For linear regression with 1 feature, just 2 examples.
  - Fit the model based on the samples.
    - Fit a line to these 2 points.
    - With ‘d’ features, you’ll need ‘d+1’ examples.
  - Test how many points are fit well based on the model.
  - Repeat until we find a model that fits at least the expected number of “inliers”.
- You might then re-fit based on the estimated “inliers”.



https://en.wikipedia.org/wiki/Random_sample_consensus

Log-Sum-Exp for Brittle Regression

- To use log-sum-exp for brittle regression:

$$\|Xw - y\|_\infty = \max_i \{|w^T x_i - y_i|\}$$

$$= \max_i \{\max\{w^T x_i - y_i, y_i - w^T x_i\}\} \quad \text{since } |z| = \max\{z, -z\}$$

$$= \log\left(\sum_{i=1}^{n} \exp(w^T x_i - y_i) + \sum_{i=1}^{n} \exp(y_i - w^T x_i)\right)$$

using log-sum-exp to approximate "max" over $2n$ terms.

Log-Sum-Exp Numerical Trick

- Numerical problem with log-sum-exp is that $\exp(z_i)$ might overflow.
  - For example, exp(100) has more than 40 digits.
- Implementation 'trick': Let $\beta = \max_i \{z_i\}$

$$\log\left(\sum_i \exp(z_i)\right) = \log\left(\sum_i \exp(z_i - \beta + \beta)\right)$$

$$= \log\left(\sum_i \exp(z_i - \beta)\exp(\beta)\right)$$

$$= \log\left(\exp(\beta)\sum_i \exp(z_i - \beta)\right)$$

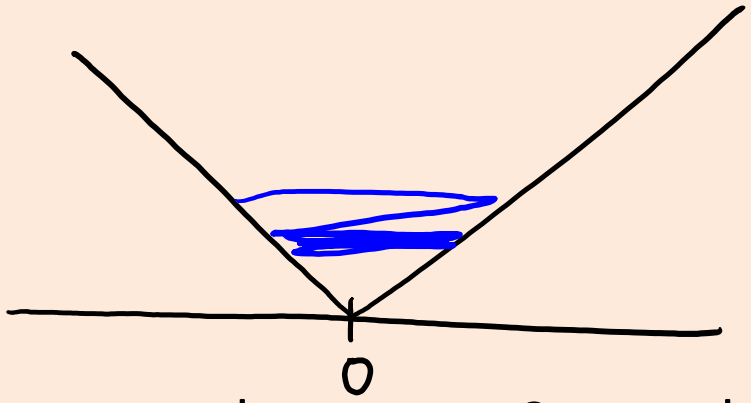
$$= \log(\exp(\beta)) + \log\left(\sum_i \exp(z_i - \beta)\right)$$

$$= \beta + \log\left(\sum_i \exp(z_i - \beta)\right)$$

$\exp(z_i - \beta) \leq 1$ so no overflow

Gradient Descent for Non-Smooth?

- "You are unlikely to land on a non-smooth point, so gradient descent should work for non-smooth problems?"
  - Consider just trying to minimize the absolute value function:



  - Norm(gradient) is constant when not at 0, so unless you are lucky enough to hit exactly 0, you will just bounce back and forth forever.
  - We didn't have this problem for smooth functions, since the gradient gets smaller as you approach a minimizer.
  - You could fix this problem by making the step-size slowly go to zero, but you need to do this carefully to make it work, and the algorithm gets much slower.

Gradient Descent for Non-Smooth?

- Counter-example from Bertsekas' "Ngradient descent for a non-smooth convex problem does not converge to a minimum.onlinear Programming" where

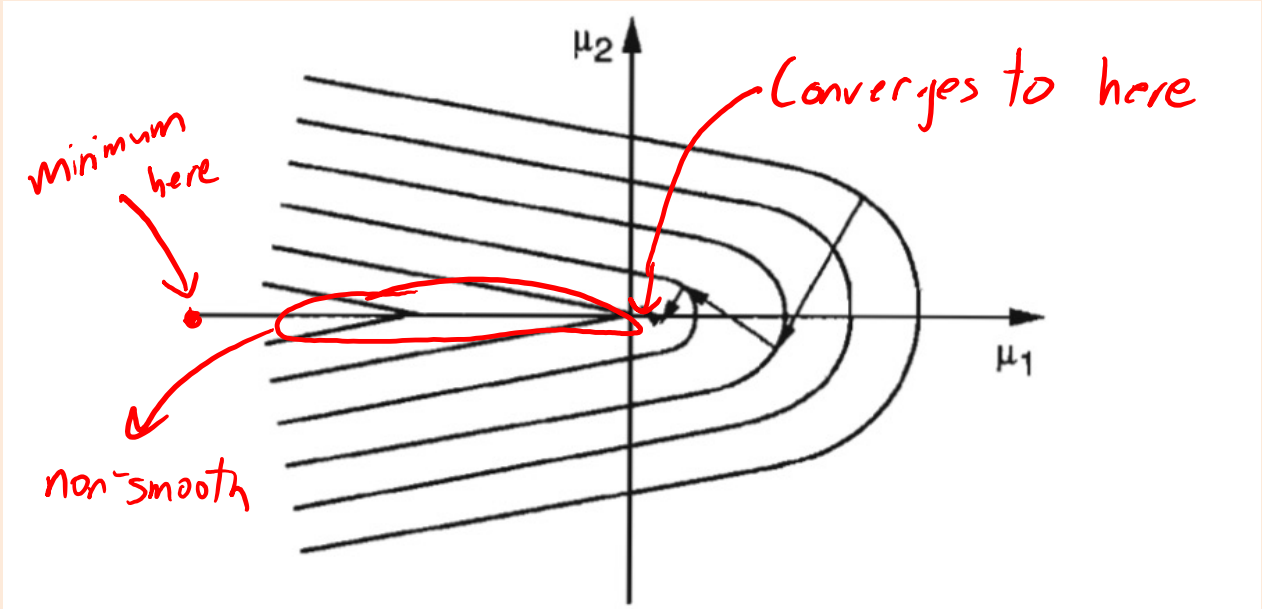


Figure 6.3.8. Contours and steepest ascent path for the function of Exercise 6.3.8.