CPSC 340:
Machine Learning and Data Mining

Robust Regression

Spring 2022 (2021W2)

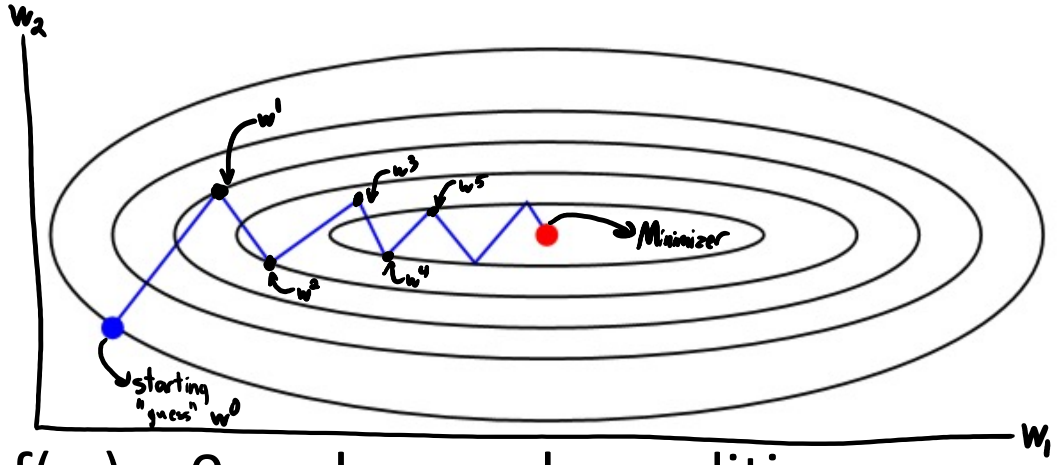
Admin

- Midterm
  - Thu Feb 17 from 6:00-7:30pm
  - You will have 85 minutes in that 90-minute window
  - Covers assignments 1-3; lectures L1 to L15 (be taught on Monday 14^{th})

- We released practice exams (on Piazza).

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 ∇ 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 constant, maximization (max of convex functions is a convex fuction).
  - Any [squared-]norm is convex.
  - Composition of convex function with linear function is convex.

Example: Convexity of Linear Regression (Easy Way)

- Consider linear regression objective with squared error:

$$f(w) = \|Xw - y\|^2$$

- We can use that this is a convex function composed with linear:

Let $h(w) = Xw - y$, which is a linear function ('d' inputs, 'n' outputs)

Let $g(r) = \|r\|^2$, which is convex because it's a squared norm.

Then $f(w) = g(h(w))$, which is convex because it's a convex function composed with a linear function

bonus!

Convexity in Higher Dimensions

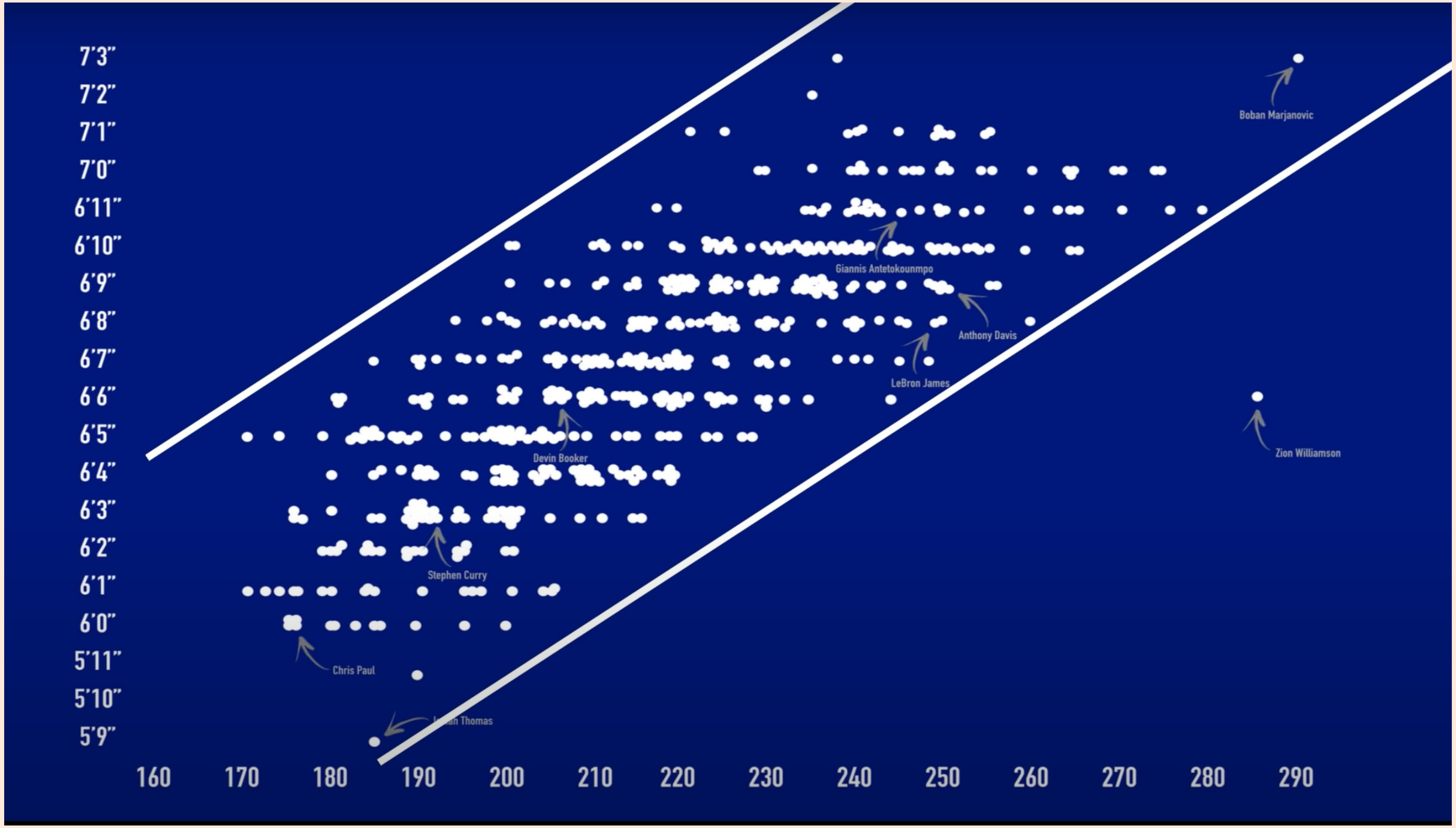
- Twice-differentiable 'd'-variable function is convex iff:
  - Eigenvalues of Hessian $\nabla^2 f(w)$ are non-negative for all 'w'.
- True for least squares where $\nabla^2 f(w)$ = X^TX for all 'w'.
  - See bonus slides for why X^TX has non-negative eigenvalues.
- Unfortunately, sometimes it is hard to show convexity this way.
  - Usually easier to just use some of the rules as we did on the last slide.

(pause)

bonus!

Least Squares with Outliers

- Height vs. weight of NBA players:

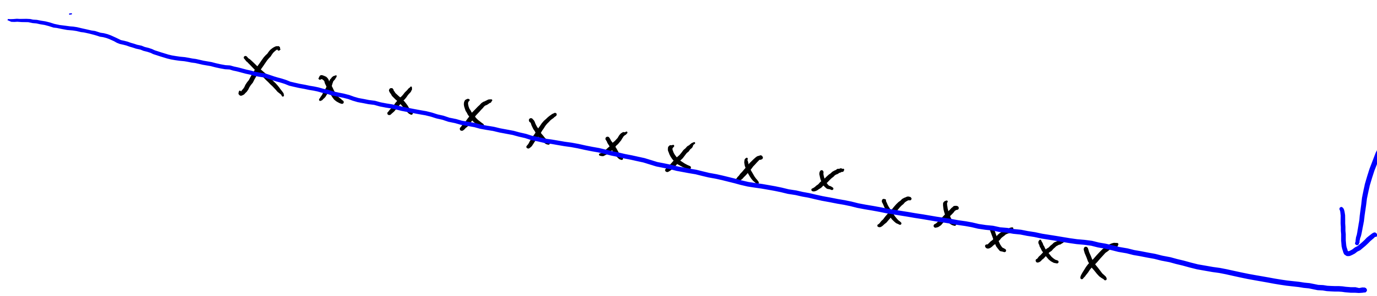


https://www.youtube.com/watch?v=i4eYWl1ewFo

Least Squares with Outliers

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

x ← "outlier" that doesn't follow trend



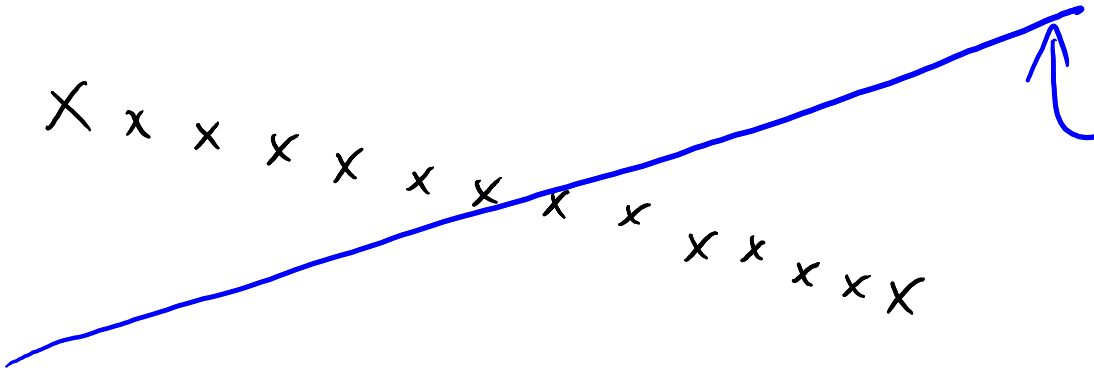
This is what we might <u>want</u> least squares to do.

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

Least Squares with Outliers

- Consider least squares problem with outliers in ‘y’:

x ← "outlier" that doesn't follow trend

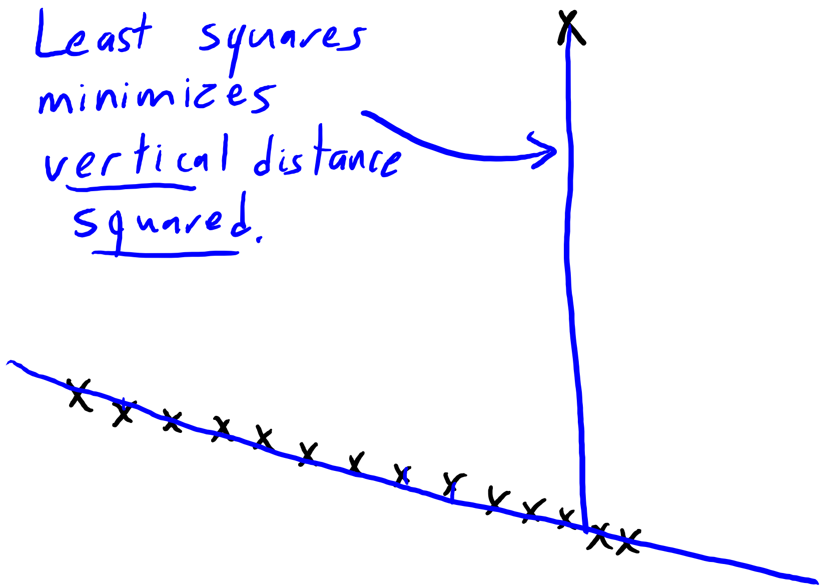


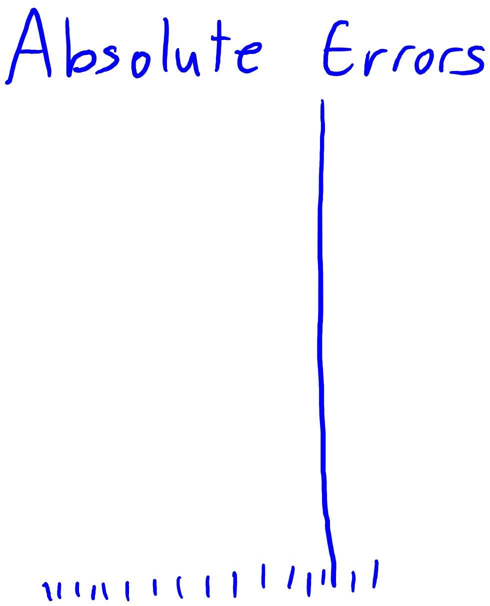
This is what least squares will actually do.

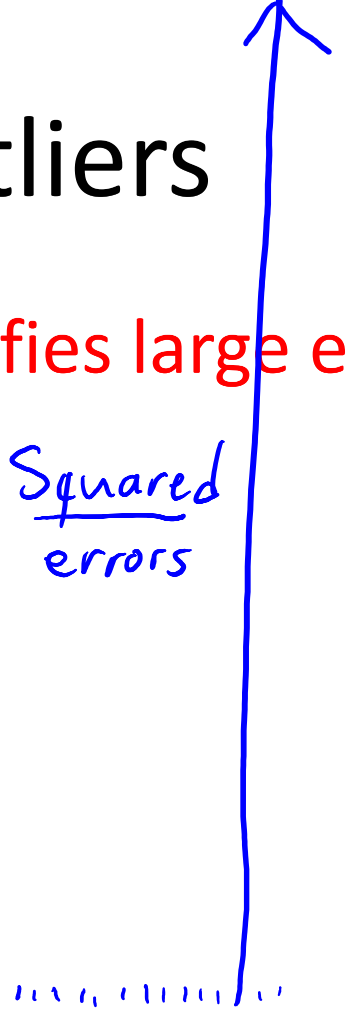
- Least squares is very sensitive to outliers.

Least Squares with Outliers

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





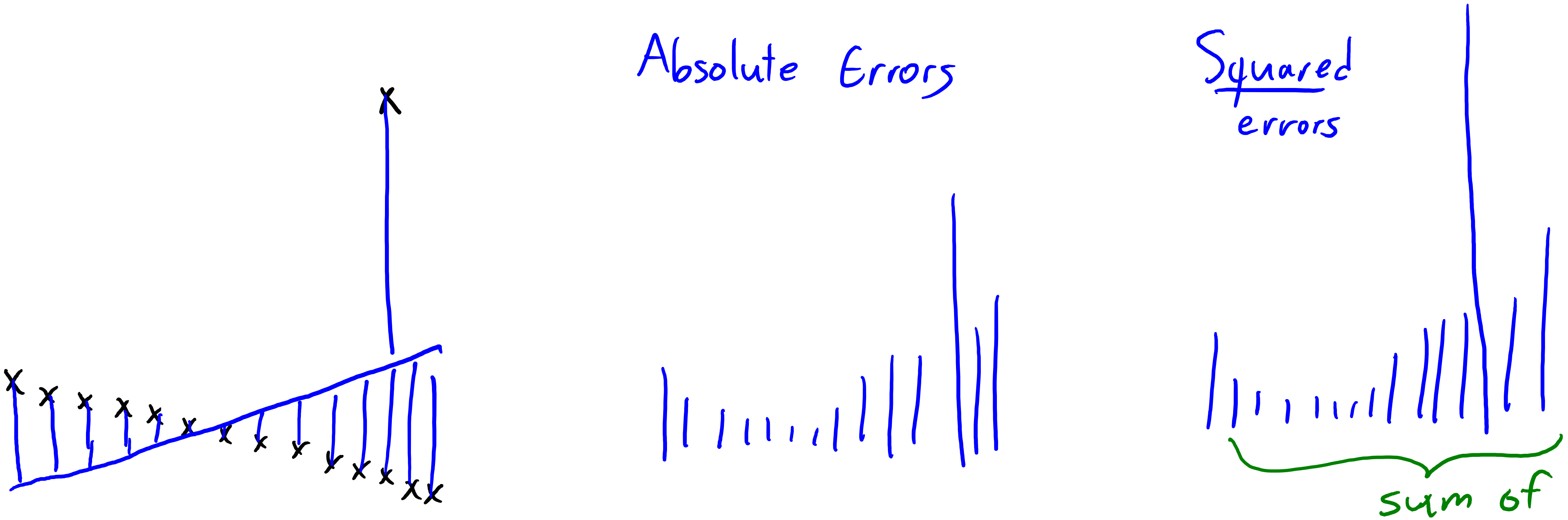


- Outliers (large error) influence ‘w’ much more than other points.

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

Least Squares with Outliers

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



sum of these is smaller than for the "correct" line.

- 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 squares:

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

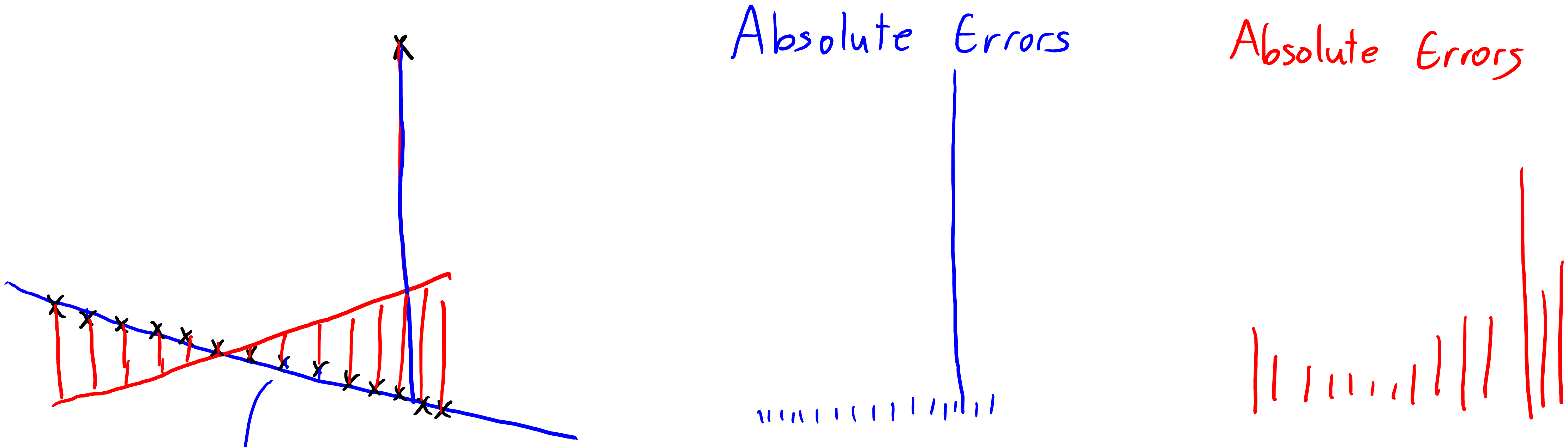
Least absolute error:

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

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

Least Squares with Outliers

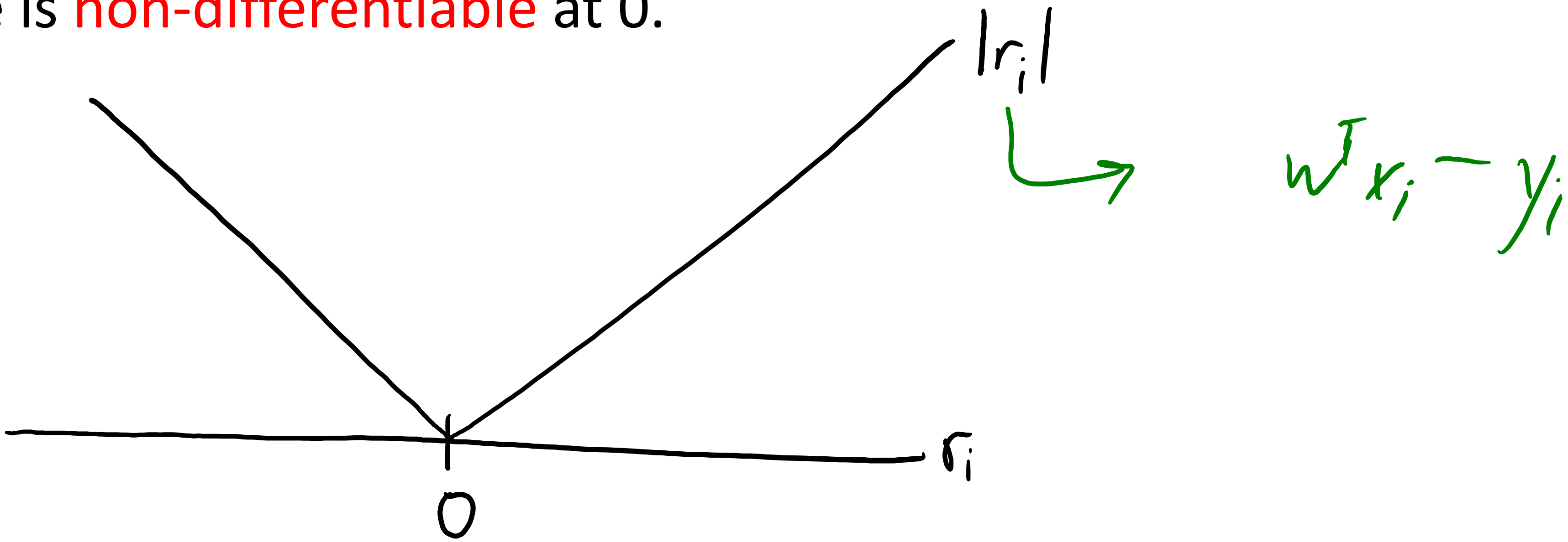
- Absolute error is more robust to outliers:



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

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.



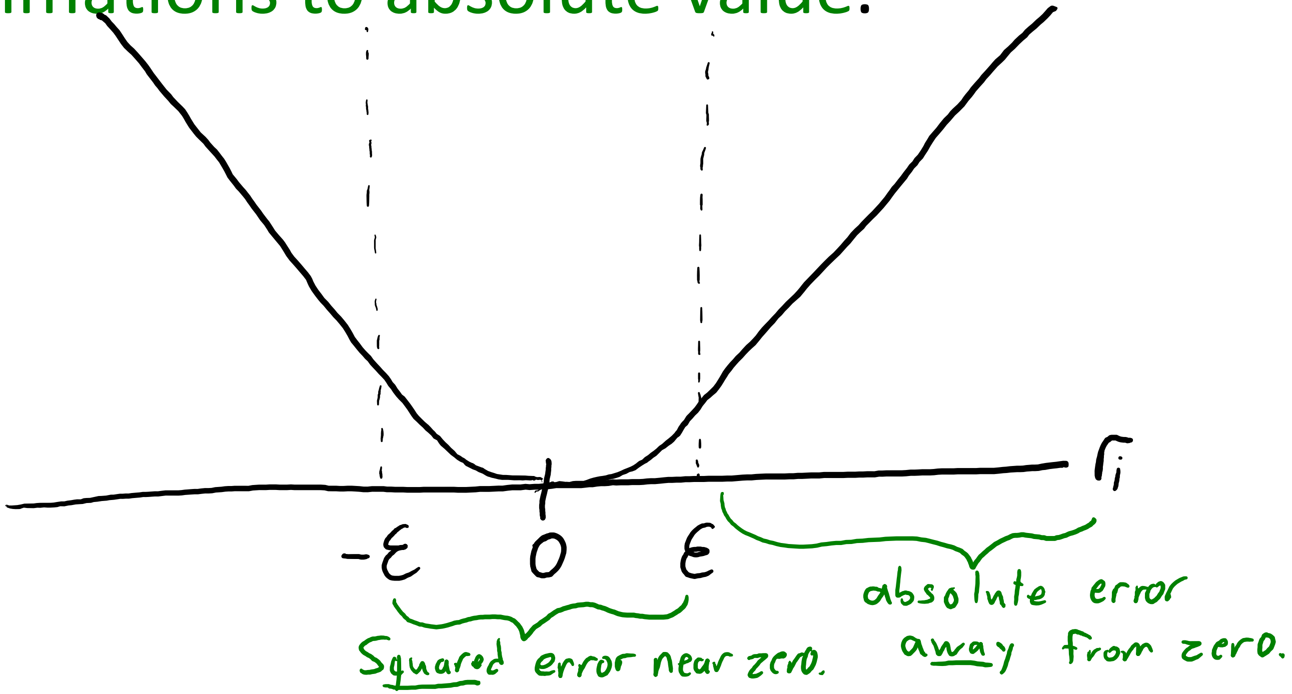
  - Generally, harder to minimize non-smooth than smooth functions.
    - Unlike smooth functions, the gradient may not get smaller near a minimizer.
  - To apply gradient descent, we’ll use a smooth approximation.

Smooth Approximations to the L1-Norm

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

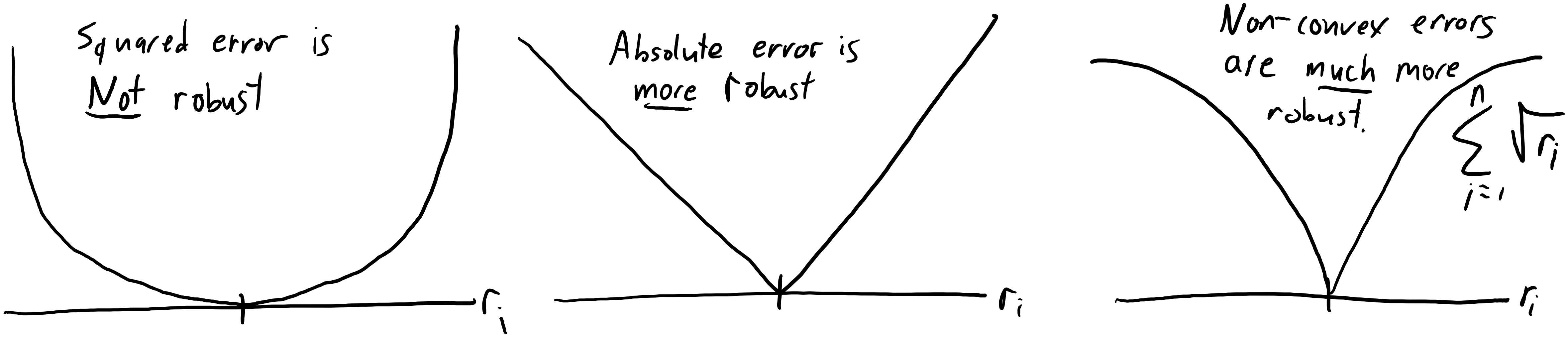
$$f(w) = \sum_{i=1}^{n} h(w^T x_i - y_i)$$

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

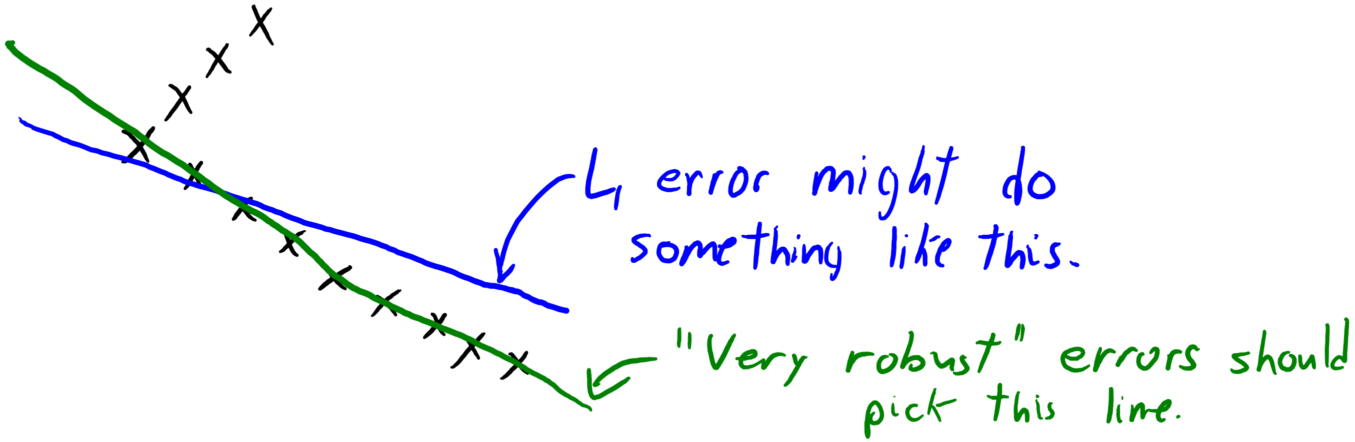


  - Note that 'h' is differentiable: $h'(\epsilon) = \epsilon$ and $h'(-\epsilon) = -\epsilon$.
  - This 'f' is convex but setting $\nabla f(x) = 0$ does not give a linear system.
    - But we can minimize the Huber loss using gradient descent.

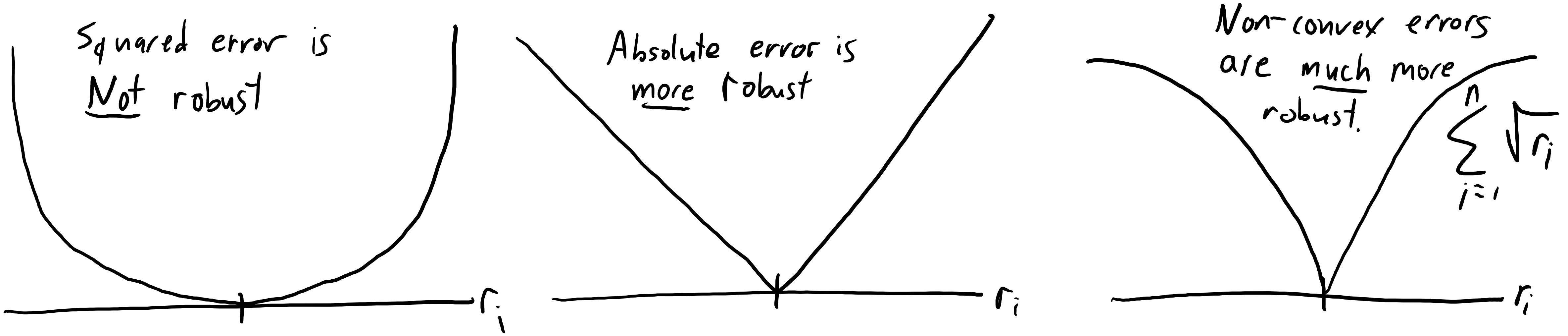
Very Robust Regression



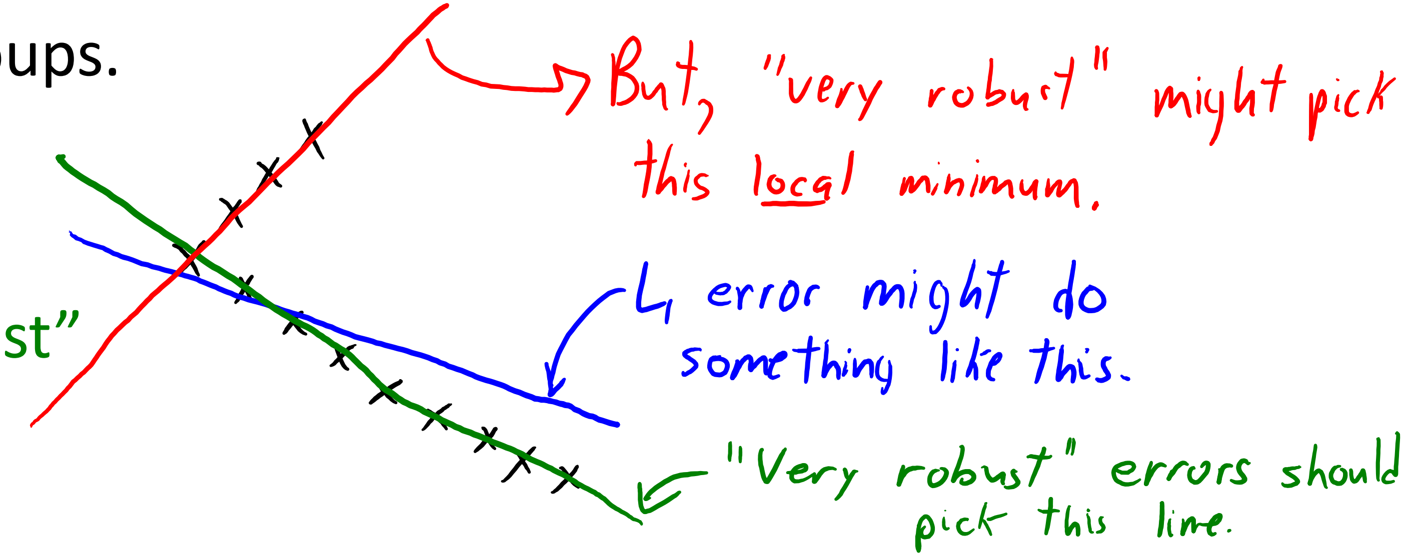
- Non-convex errors can be very robust:
  - Not influenced by outlier groups.



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.



bonus!

Motivation for Modeling Outliers

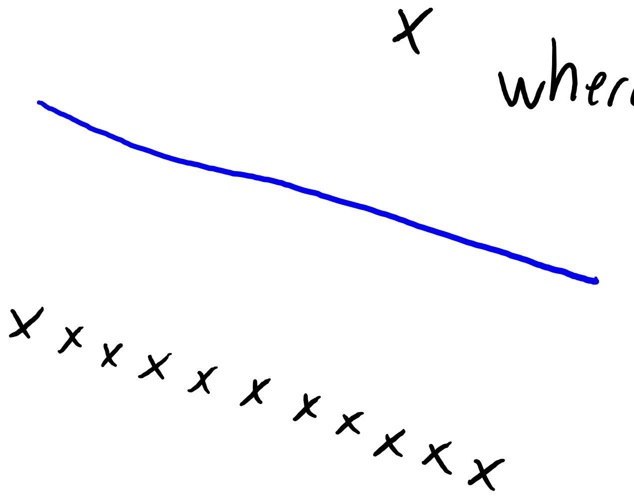
- What if the “outlier” is the only non-male person in your dataset?
  - Do you want to be robust to the outlier?
  - Will the model work for everyone if it has good average case performance?

https://xkcd.com/937/

"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.
- In this case you could use something like the infinity-norm:

$$f(w) = \|Xw - y\|_\infty \quad \text{where} \quad \|r\|_\infty = \max_i \{|r_i|\}$$



- Very sensitive to outliers ("brittle"), but minimizes worst (highest) errors.

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'll use this several times in the course.
  - Notation reminder: 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:

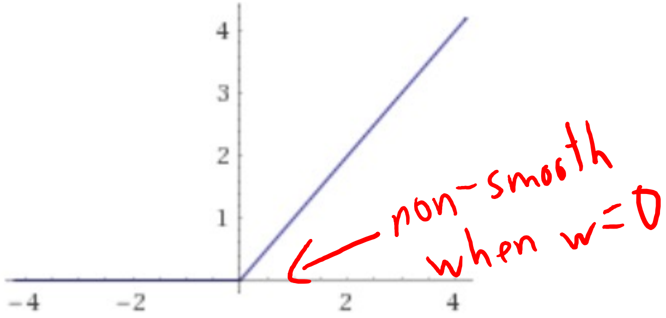
$$\max_i \{z_i\} \approx \log\left(\sum_i \exp(z_i)\right)$$

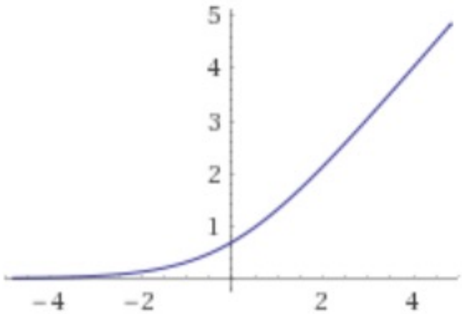
- If there aren't "close" values, it's really close to the max.

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 $z_i = \{2, 20, 19.99, -100, 7\}$ 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)):





Recap of Part 3

Linear Models, Least Squares

- Focus of Part 3 is linear models:
  - Supervised learning where prediction is linear combination of features:

$$\hat{y}_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}$$
$$= w^T x_i$$

- Regression:
  - Target y_i is numerical, testing ($\hat{y}_i == y_i$) doesn't make sense.

Good fit that doesn't exactly pass through any point.

- Squared error: $\frac{1}{2}\sum_{i=1}^{n}(w^T x_i - y_i)^2$ or $\frac{1}{2}\|Xw - y\|^2$
  - Can find optimal 'w' by solving "normal equations".

Change of Basis, Gradient Descent

- Change of basis: replaces features x_i with non-linear transforms z_i:
  - Add a bias variable (feature that is always one).
  - Polynomial basis.
  - Other basis functions (logarithms, trigonometric functions, etc.).

- For large 'd' we often use gradient descent:
  - Iterations only cost O(nd).
  - Converges to a critical point of a smooth function.
  - For convex functions, it finds a global optimum.

Error Functions, Smoothing

- Error functions:
  - Squared error is sensitive to outliers.
  - Absolute (L_1) error and Huber error are more robust to outliers.
  - Brittle (L_∞) error is more sensitive to outliers.
- L_1 and L_∞ error functions are convex but non-differentiable:
  - Finding 'w' minimizing these errors is harder than squared error.
- We can approximate these with differentiable functions:
  - L_1 can be approximated with Huber.
  - L_∞ can be approximated with log-sum-exp.
- With these smooth (convex) approximations,
  we can find global optimum with gradient descent.

Finding the "True" Model

- What if our goal is 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 way too large.
    (training error always decreases as you increase 'p')

Finding the "True" Model

- What if our goal is 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.

  - Even if true model has p=2, this is a special case of a degree-3 polynomial.
  - If 'p' is too big then we overfit, but might still 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 $Z_p = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^p \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^p \\ 1 & x_3 & (x_3)^2 & \cdots & (x_3)^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & (x_n)^2 & \cdots & (x_n)^p \end{bmatrix}$

Find 'p' that minimizes:

$$\text{score}(p) = \underbrace{\frac{1}{2}\|Z_p v - y\|^2}_{\text{train error for best 'v' with this basis.}} + \underbrace{p}_{\text{degree of polynomial}}$$

  - 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?

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

  - Form Z_0, solve for 'v', compute score(0) = $\frac{1}{2}||Z_0v - y||^2 + 0$.
  - Form Z_1, solve for 'v', compute score(1) = $\frac{1}{2}||Z_1v - y||^2 + 1$.
  - Form Z_2, solve for 'v', compute score(2) = $\frac{1}{2}||Z_2v - y||^2 + 2$.
  - Form Z_3, solve for 'v', compute score(3) = $\frac{1}{2}||Z_3v - y||^2 + 3$.

  - 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 λ > 0 controls how strong we penalize complexity.
    - "You need to decrease the training error by least λ to increase 'k' by 1".

- Using (λ = 1) is called Akaike information criterion (AIC).
- Other choices of λ (not necessarily integer) give other criteria:
  - Mallow's C_p.
  - Adjusted R^2.
  - ANOVA-based model selection.

bonus!

Naming something after yourself without being gauche

Akaike Information Criterion

Watanabe-Akaike Info. Criterion

P.S. When introducing AIC, Akaike called it An Information Criterion. When introducing WAIC, Watanabe called it the Widely Applicable Information Criterion. Aki and I are hoping to come up with something called the Very Good Information Criterion.

Aki Vehtari

Andrew Gelman

https://statmodeling.stat.columbia.edu/2013/08/09/understanding-predictive-information-criteria-for-bayesian-models/

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.

Summary

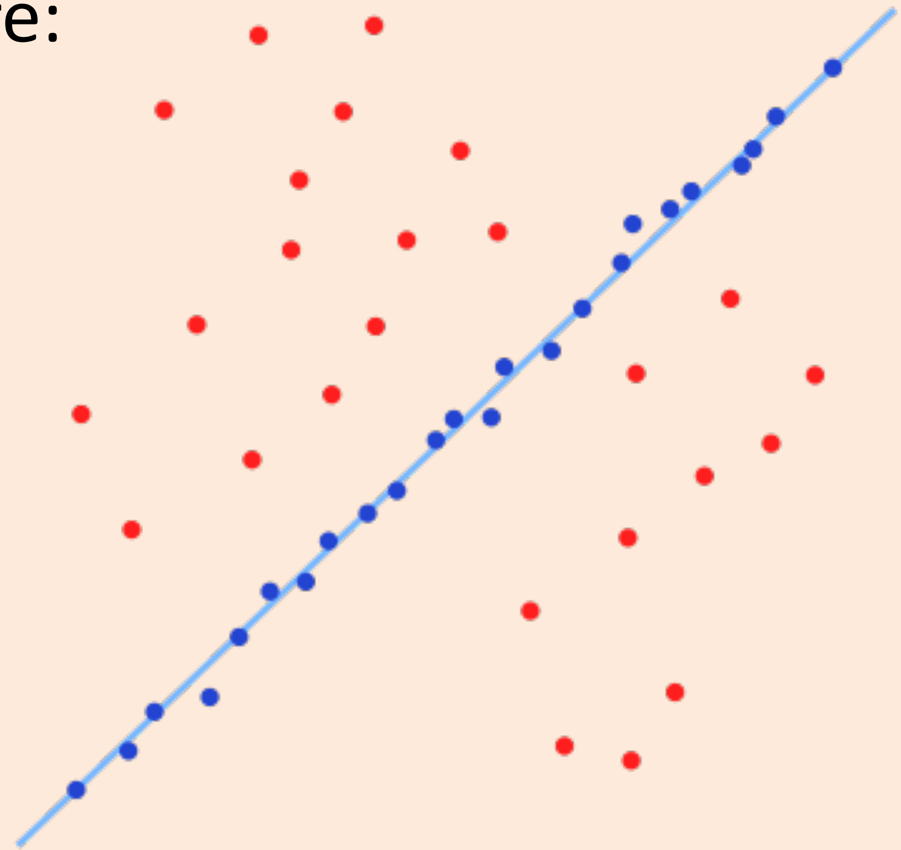
- Outliers in ‘y’ can cause problem for least squares.
- Robust regression using L1-norm is less sensitive to outliers.
- Brittle regression using Linf-norm is more sensitive to outliers.
- Smooth approximations:
  - Let us apply gradient descent to non-smooth functions.
  - Huber loss is a smooth approximation 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?

bonus!

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.

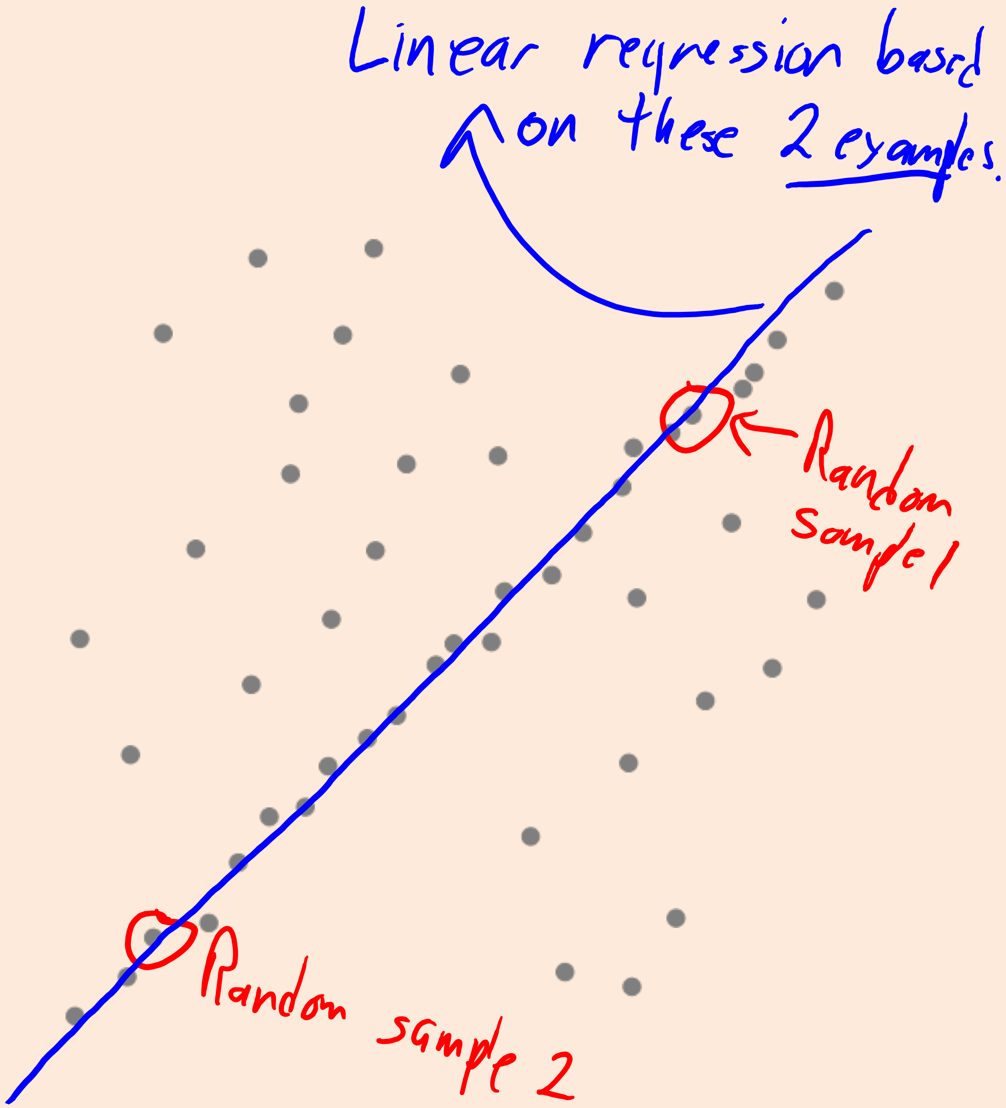


https://en.wikipedia.org/wiki/Random_sample_consensus

bonus!

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”.



bonus!

Log-Sum-Exp for Brittle Regression

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

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

$$= \max_i \{\max\{w^T x_i - y_i, y_i - w^T x_i\}\}$$ 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.

bonus!

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 $\beta = \max_i \{z_i\}$

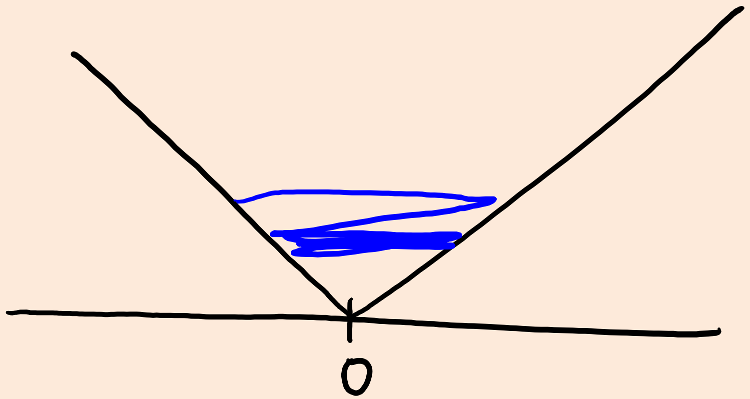
$$
\begin{aligned}
\log\left(\sum_i \exp(z_i)\right) &= \log\left(\sum_i \exp(z_i - \beta + \beta)\right) \\
&= \log\left(\sum_i \exp(z_i - \beta)\exp(\beta)\right) \\
&= \log\left(\exp(\beta)\sum_i \exp(z_i - \beta)\right) \\
&= \log(\exp(\beta)) + \log\left(\sum_i \exp(z_i - \beta)\right) \\
&= \beta + \log\left(\sum_i \exp(z_i - \beta)\right)
\end{aligned}
$$

≤ 1 so no overflow

bonus!

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' "Nonlinear Programming" where gradient descent for a non-smooth convex problem does not converge to a minimum.

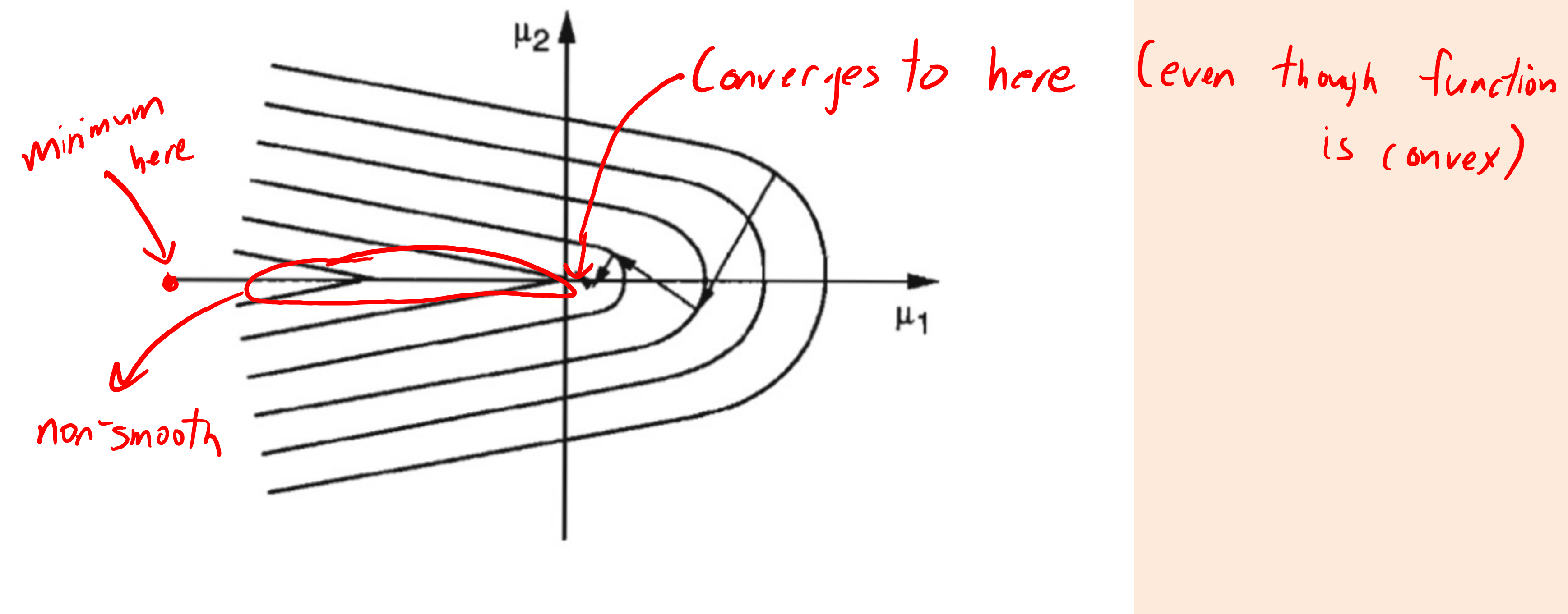


Figure 6.3.8. Contours and steepest ascent path for the function of Exercise 6.3.8.

bonus!

Example: Convexity of Linear Regression (Hard Way)

- Consider linear regression objective with squared error:

$$f(w) = \|Xw - y\|^2$$

- Twice-differentiable ‘f’ is convex if $\nabla^2 f(x)$ has eigenvalues ≥ 0.
  - This is equivalent to saying $v^T \nabla^2 f(x) v \geq 0$ for all vectors v.
- The Hessian for least squares is $\nabla^2 f(x) = X^T X$.
  - See notes on Gradients and Hessians of quadratics on webpage.
- We have: $v^T \nabla^2 f(w) v = v^T X^T X v = (Xv)^T (Xv) = \|Xv\|^2 \geq 0$ (because norms are ≥ 0)

So it's convex