CPSC 340: Machine Learning and Data Mining

Robust Regression

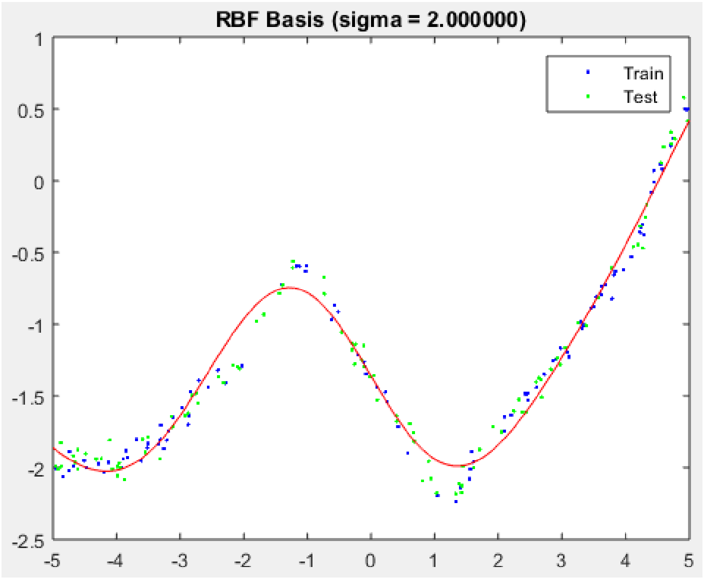
Fall 2015

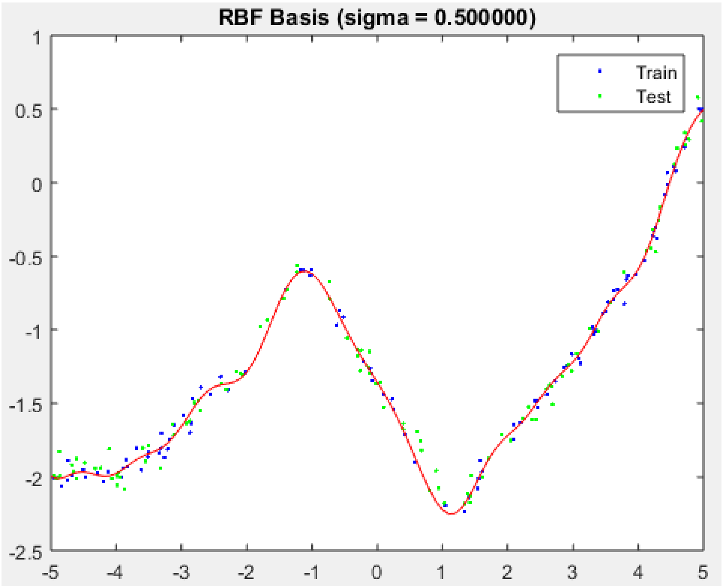
Admin

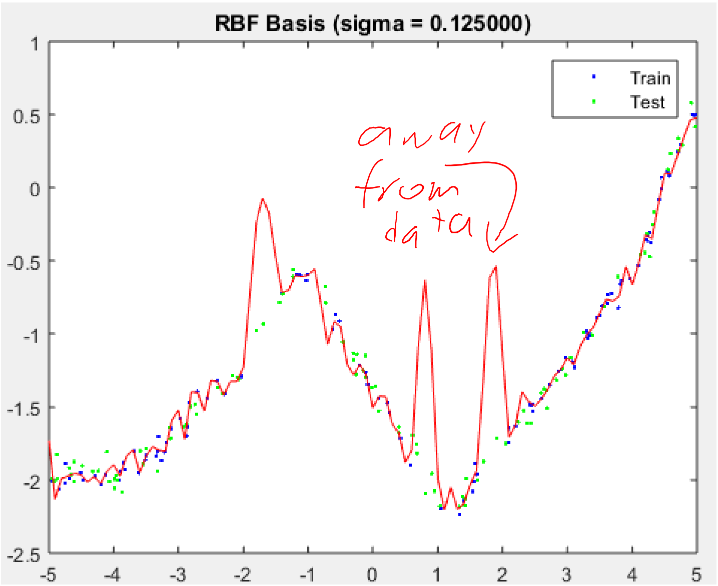
- Can you see Assignment 1 grades on UBC connect?
  – Auditors, don’t worry about it.
- You should already be working on Assignment 3.
- Notes regarding midterm:
  – This lecture is the last topic that the midterm will cover.
  – Practice midterm coming soon.
  – Questions will be similar to assignment questions.
  – Questions will only cover topics covered in Assignments 1-3.

RBF Basis with L2-Regularization

- Use {RBF basis, L2-Regularization} and Cross-validation for $\{\sigma, \lambda\}$
  - Non-parametric basis, magic of regularization, and tuning for test error!

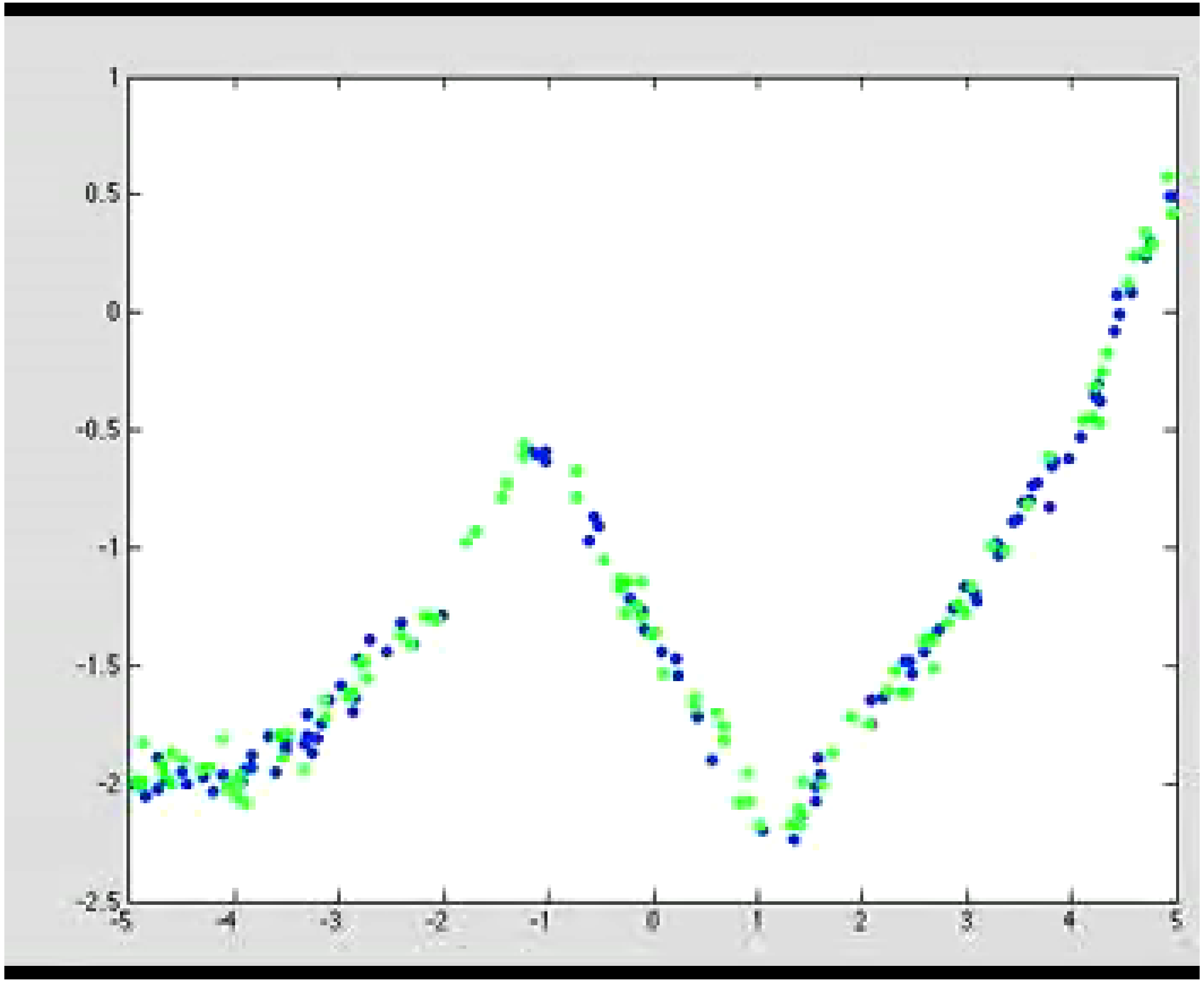




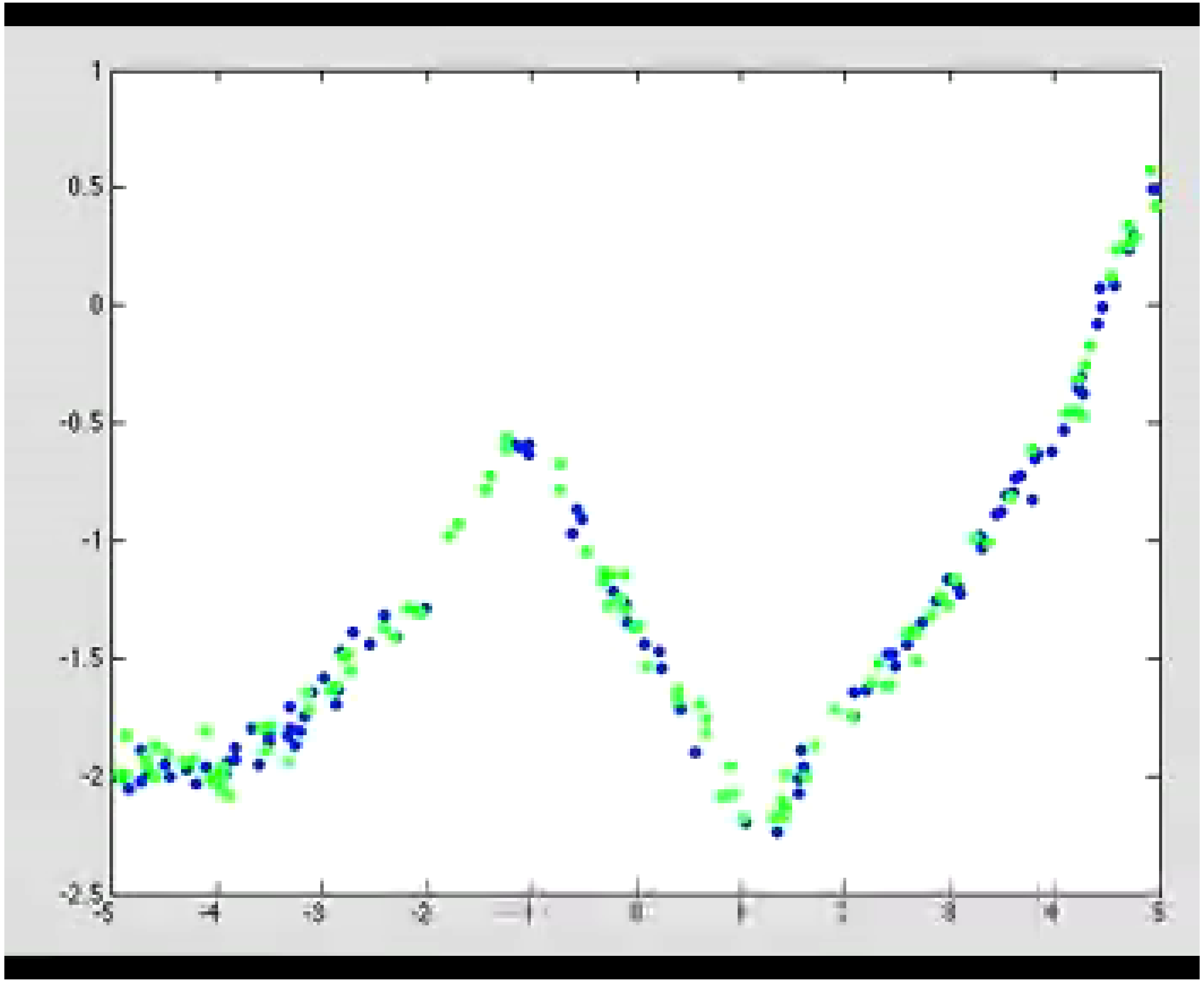


  - Can add bias or linear/poly basis to do better away from data.
  - Like KNN, it's expensive at test time.

RBF Basis with L2-Regularization

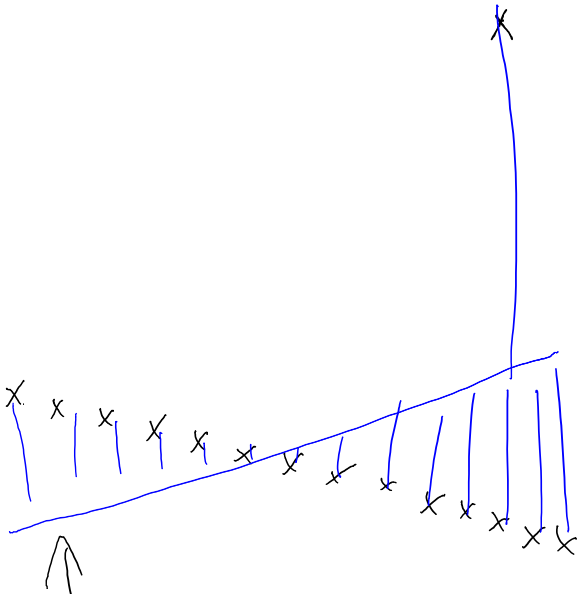


RBF Basis with L2-Regularization



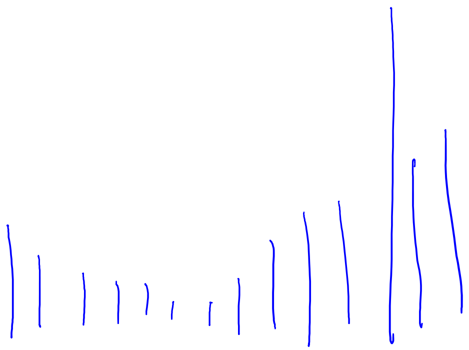
Least Squares with Outliers

- Least squares is very sensitive to outliers.

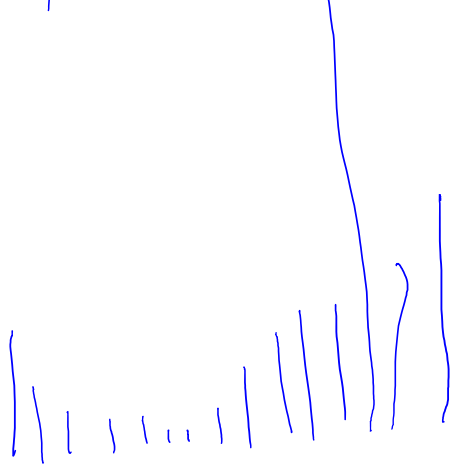


$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2}\|y - Xw\|^2$$

Absolute Errors:

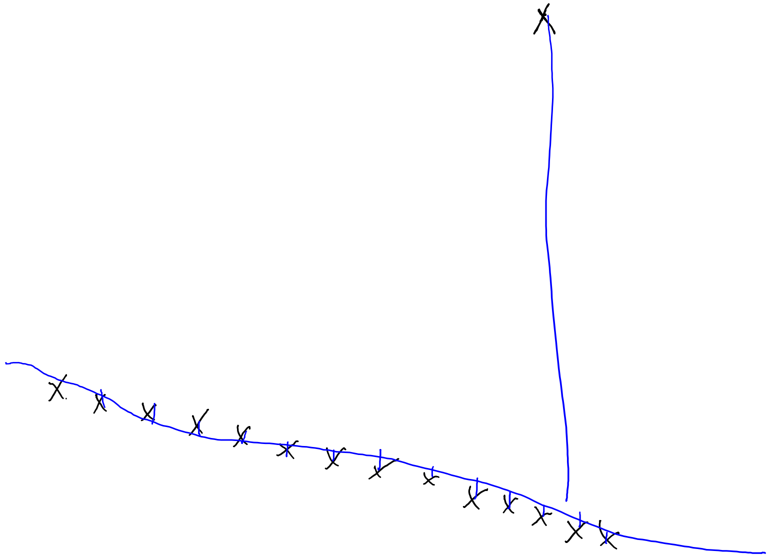


Squared Errors:

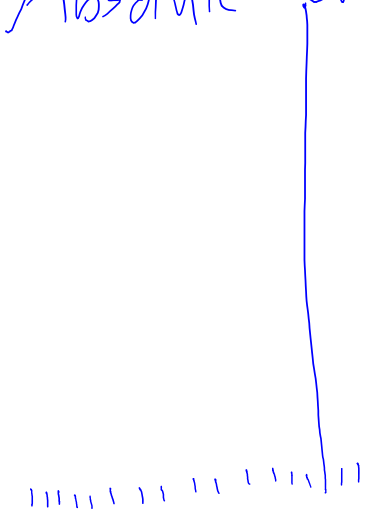


Robust Regression with Outliers

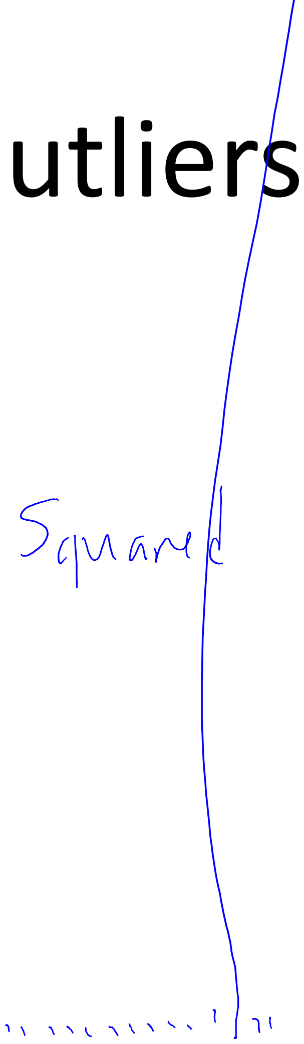
- Absolute error is more robust to outliers:



Absolute errors



Squared errors

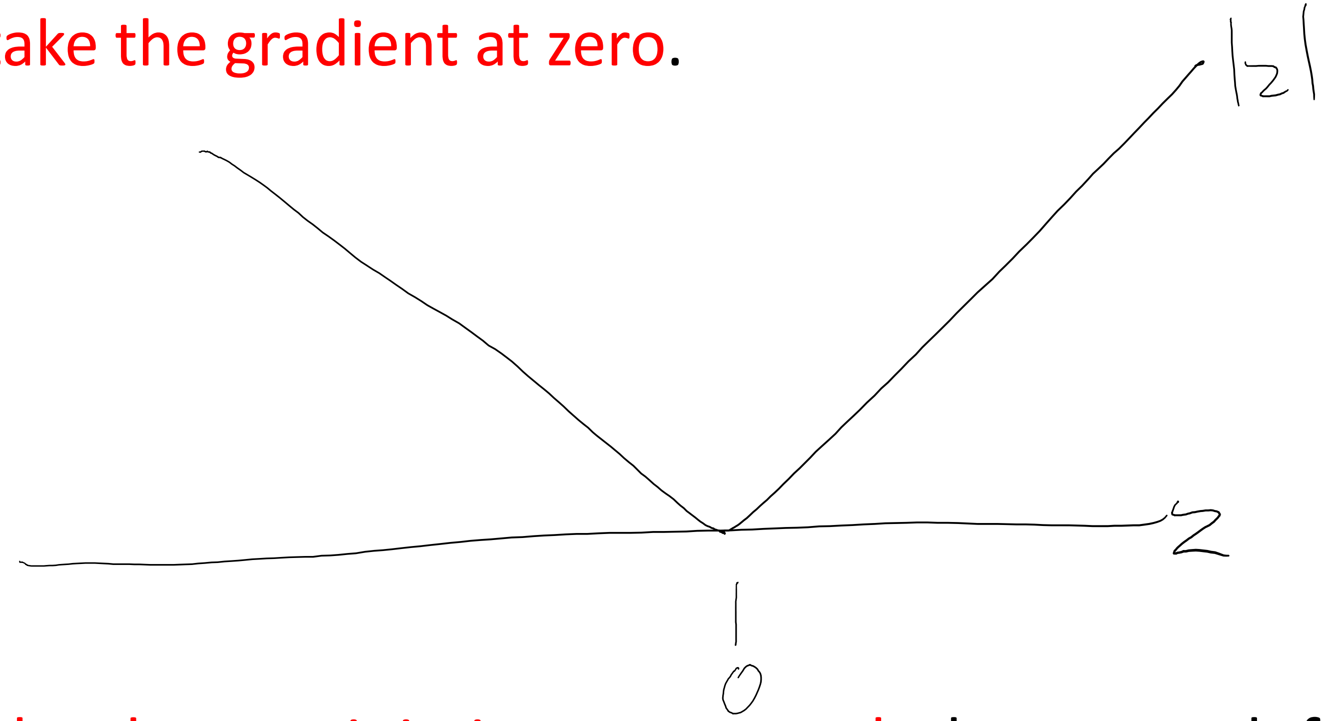


$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \|y - Xw\|_1$$

"L_1-norm": $\|z\|_1 = \sum_{i=1}^{n} |z_i|$

Regression with the L1-Norm

- Unfortunately, minimizing the absolute error is harder:
  - We can't take the gradient at zero.



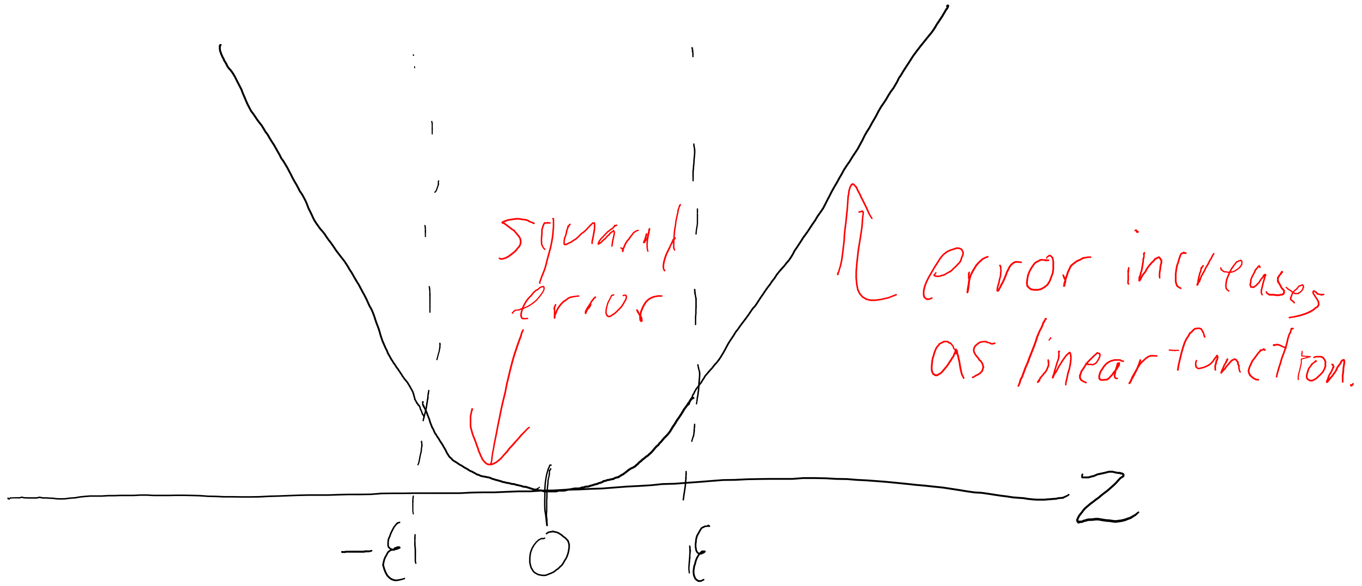
  - Generally, harder to minimize non-smooth than smooth functions.
  - Could solve as 'linear program', but harder than 'linear system'.

Smooth Approximations to the L1-Norm

- There are differentiable approximations to absolute value.
- For example, the Huber loss:

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} h(y_i - w^T y_i)$$

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

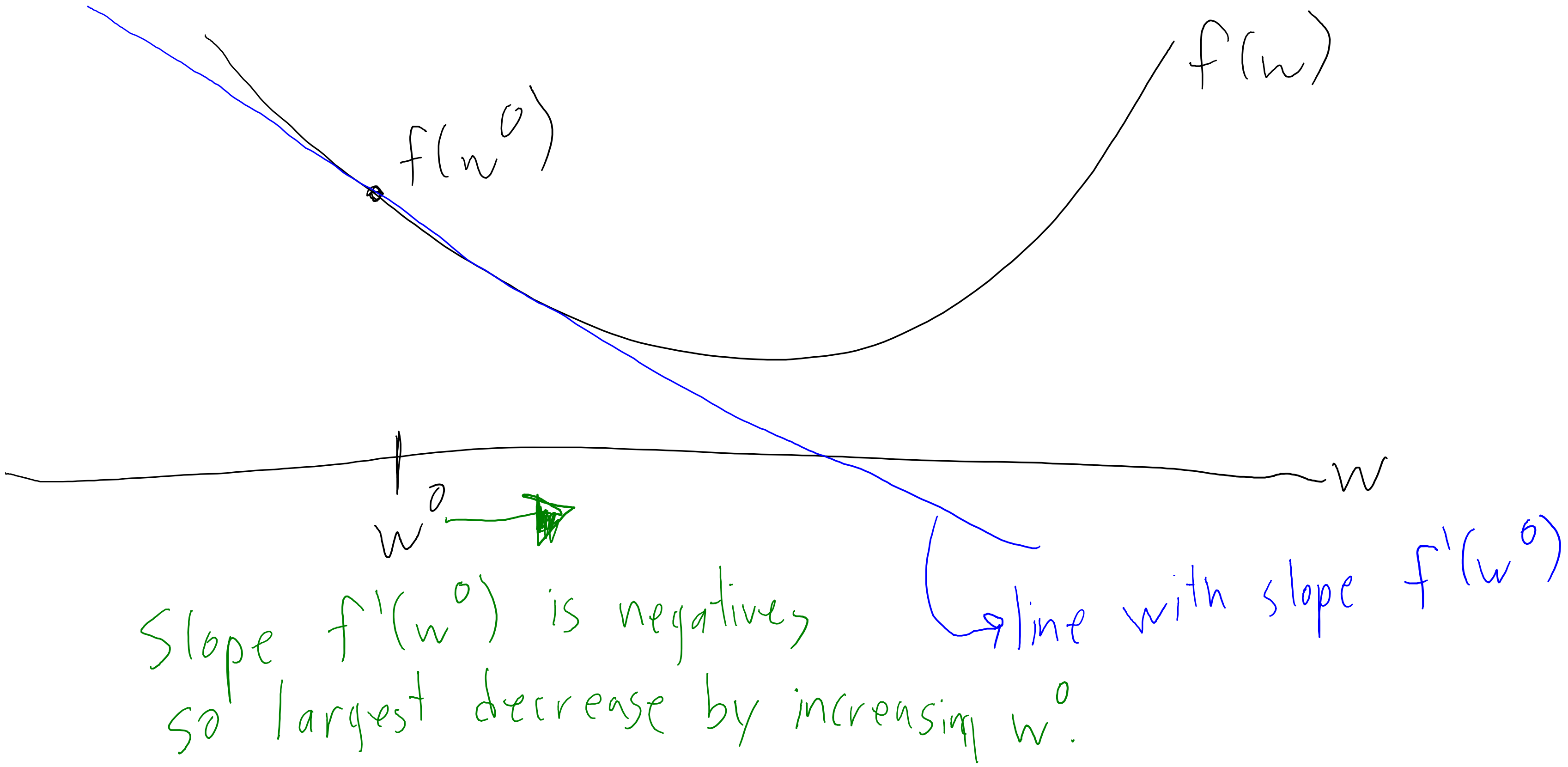


- No closed-form solution for $\nabla f(x) = 0$, but can use gradient descent.

$$\nabla f(w) = -\sum_{i=1}^{n} x_i h'(y_i - w^T x_i) \qquad h'(z) = \begin{cases} z & \text{for } |z| \leq \varepsilon \\ \varepsilon \operatorname{sign}(z) & \text{otherwise.} \end{cases}$$

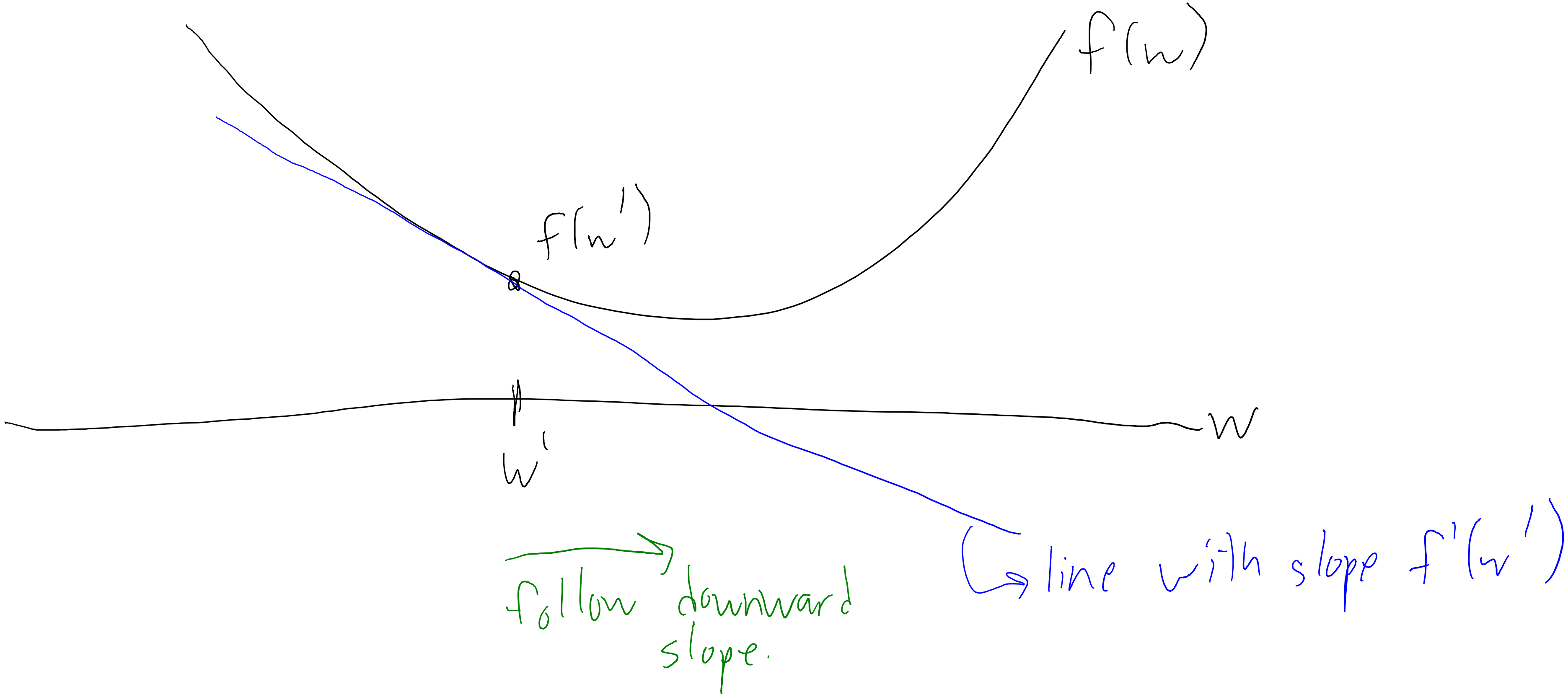
Gradient Descent

- Gradient descent is based on a simple observation:
  - Give parameters ‘w^0’, vector direction of largest decrease is $-\nabla f(w^0)$).



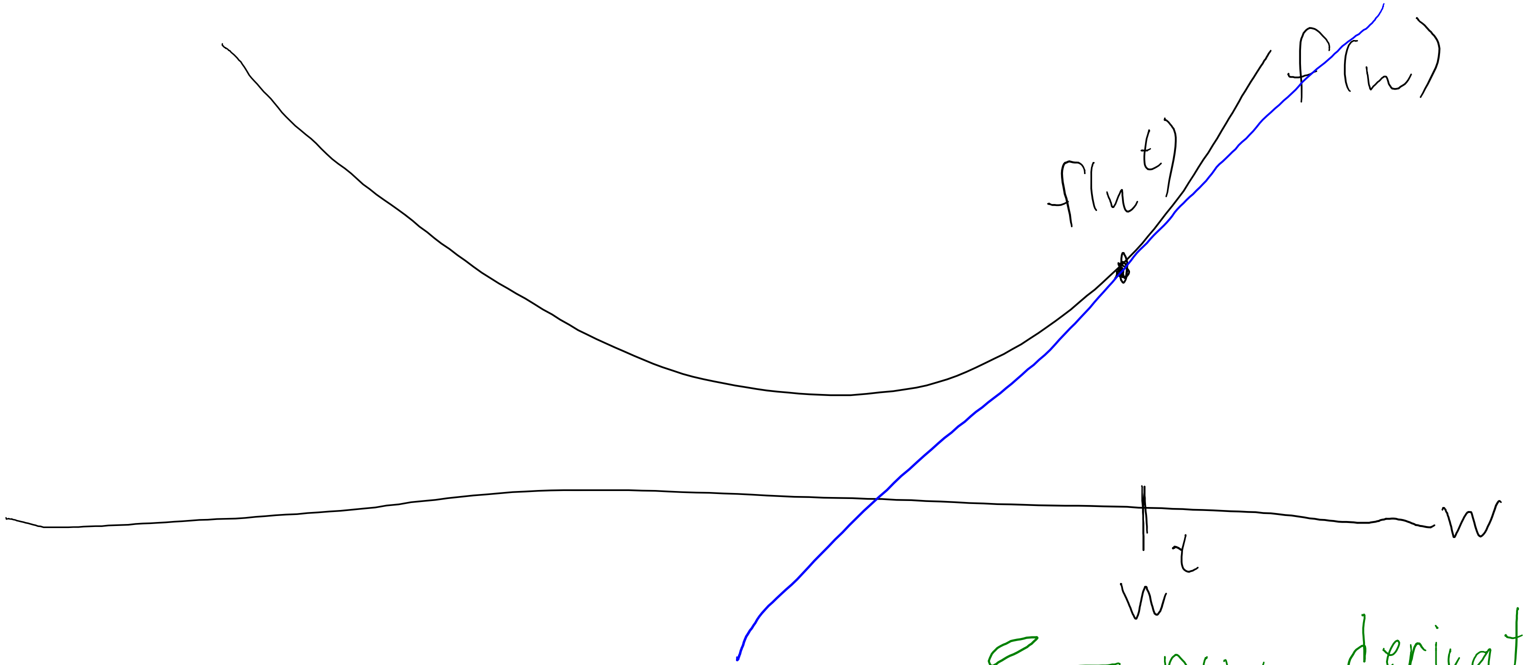
Gradient Descent

- Gradient descent is based on a simple observation:
  - Give parameters ‘w^0’, vector direction of largest decrease is $-\nabla f(w^0)$).



Gradient Descent

- Gradient descent is based on a simple observation:
  - Give parameters ‘w^0’, vector direction of largest decrease is $-\nabla f(w^0)$).



← now derivative is positive, so want to <u>decrease</u> w.

Gradient Descent

- Gradient descent is an iterative algorithm:
  - We start with some initial guess, w^0.
  - Generate new guess by moving in the negative gradient direction:

$$w^1 = w^0 - \alpha^0 \nabla f(w^0).$$

(The scalar α^0 is the `step size'.)

  - Repeat to successively refine the guess: $w^{t+1} = w^t - \alpha^t \nabla f(w^t).$

    Generate $w^2, w^3, w^4, \ldots$

  - Stop if not making progress or $\|\nabla f(w^t)\| \leq \delta$ (some small number)

Gradient Descent

$\nabla f(x) = \underbrace{X^T(Xw - y)}_{O(nd)}$

- If α^t is small enough and $\nabla f(w^k) \neq 0$, guaranteed to decrease 'f':

$$f(w^{t+1}) < f(w^t).$$

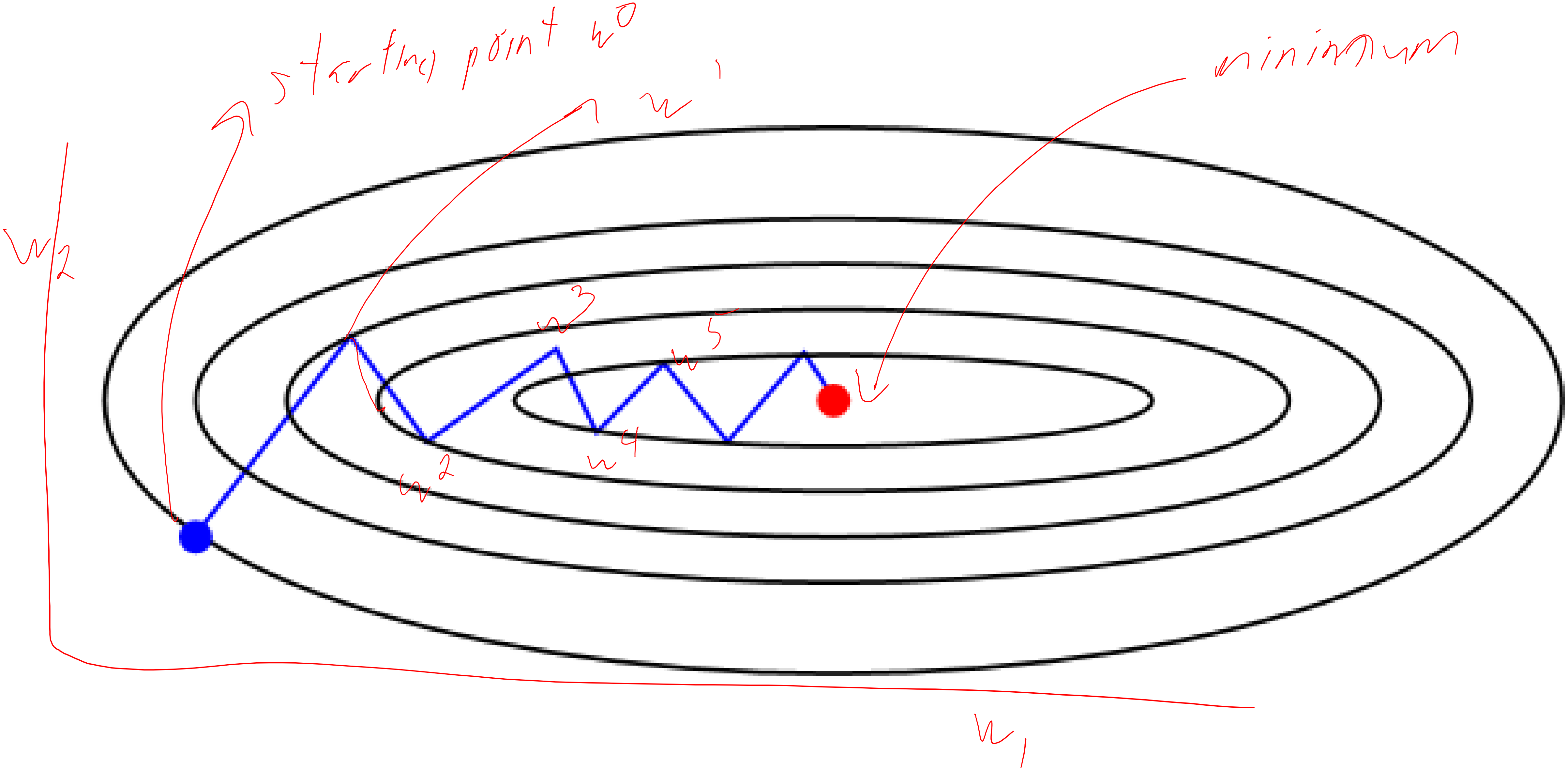
- Under weak conditions, procedure converges to a local minimum.
- Least squares via normal equations vs. gradient descent:
  - Normal equations cost $O(nd^2 + d^3)$.
  - Gradient descent costs $O(ndt)$ to run for 't' iterations.
  - If solution is good enough after t iterations, gradient descent can be faster:
    - This is true if ($t < d$) and ($t < d^2/n$), gradient descent is often better when d is very large.
- Nesterov's and Newton's methods are variants with fewer iterations.
  - For special case of L2-regularized least squares, can also use 'conjugate' gradient.

$X \in \mathbb{R}^{n \times d}$

Forming X^TX: $O(nd^2)$

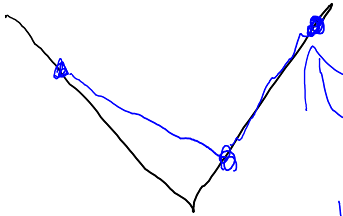
Inverting X^TX: $O(d^3)$

Gradient Descent in 2D

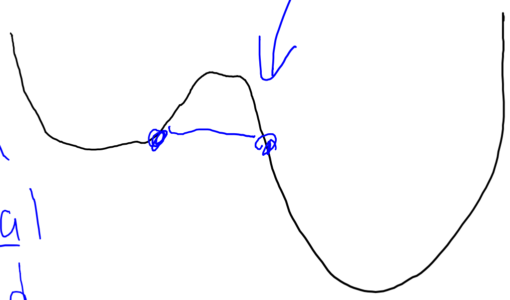


Convex Functions

- Is finding a local minimum good enough?
  - For least squares and Huber loss: yes, because they are convex.
- A function is convex if:
  - Domain is a convex set, and function is never above 'chord'.

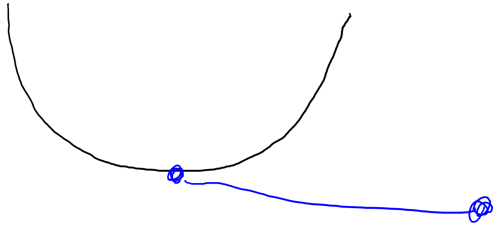


it can be equal to chord.



above chord means not convex.

  - "Curved upwards everywhere".
- All local minima of convex functions are also global minima:



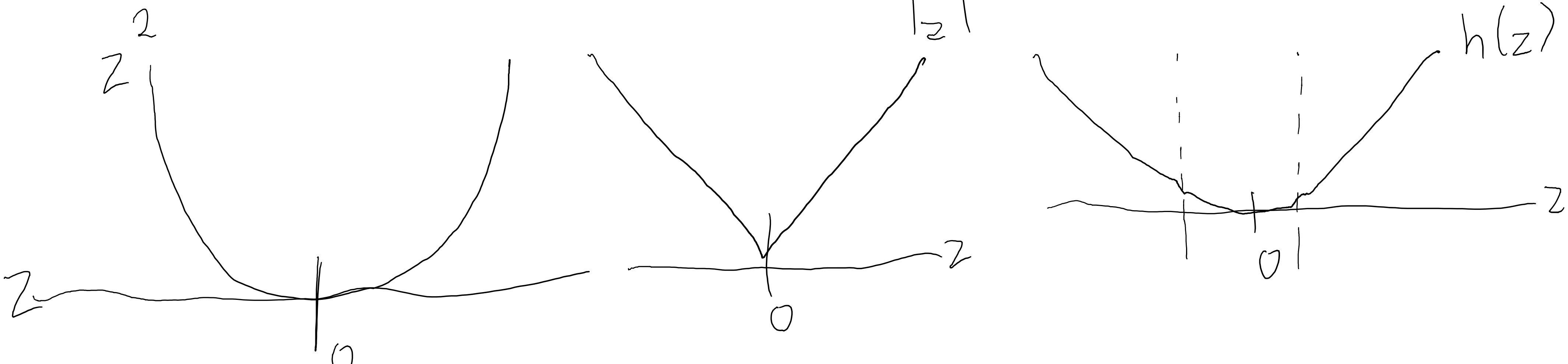
If a local minimum, it would have to curve downwards to reach lower value.

General Convex Error Functions

- Consider a general linear regression objective:

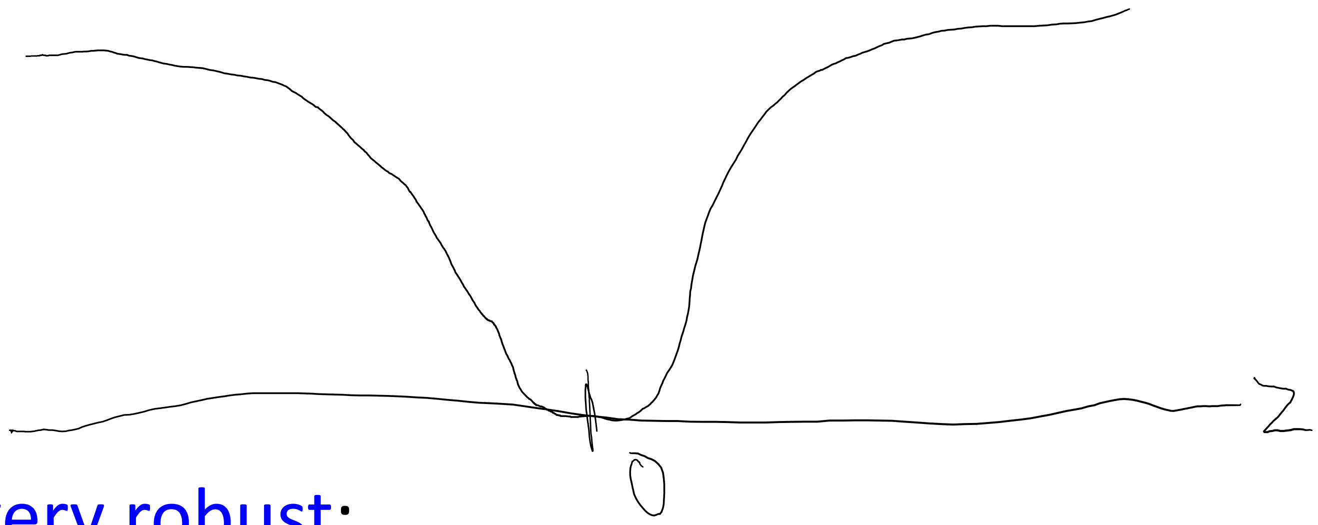
$$f(w) = \sum_{i=1}^{n} g(y_i - w^T x_i).$$

- If the ‘error’ function ‘g’ is convex, then we can show ‘f’ is convex.
- Square function, absolute value, Huber loss:



Very Robust Regression

- We could also consider non-convex or concave error functions:



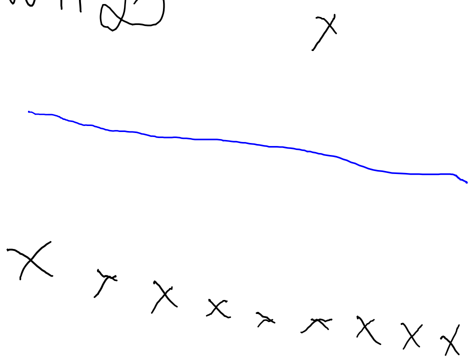
- These can be very robust:
  - Eventually 'give up' on trying to make large errors smaller.
- With non-convex errors, finding global minimum is hard.
- Absolute value is the most robust convex error function.

Motivation for Considering Worst Case

‘Brittle’ Regression

- What if you really care about getting the outliers right?
  - You want best performance on worst training example.
  - For example, if in worst case the plane can crash.
- In this case you can use something like the infinity-norm:

$$\underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \|y - Xw\|_\infty \qquad \|z\|_\infty = \max_i \{|z_i|\}$$



- Very sensitive to outliers (brittle), but worst case will be better.

Log-Sum-Exp Function

- As with the L1-norm, the L∞-norm is convex but non-smooth.
  - True for all norms (recall that we always square the L2-norm).
- Log-Sum-Exp function is a smooth approximation to max function:

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

- Intuition:
  - $\sum_i \exp(x_i) \approx \max_i \exp(x_i)$, largest element is magnified exponentially.
  - Recall that $\log(\exp(x_i)) = x_i$.
- To use for brittle regression: $\|y - Xw\|_\infty = \max_i \{|y_i - w^T x_i|\}$

$$= \max_i \{\max\{y_i - w^T x_i, w^T x_i - y_i\}\} = \log\left(\sum_i \exp(y_i - w^T x_i) + \sum_i \exp(w^T x_i - y_i)\right)$$

Log-Sum-Exp Trick

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

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

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

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

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

$$= \beta + \log\left(\sum_i \underbrace{\exp(x_i - \beta)}_{\leq 1}\right)$$

Summary

- Robust regression using L1-norm/Huber is less sensitive to outliers.
- Gradient descent finds local minimum of differentiable function.
- Convex functions do not have non-global local minima.
- Log-Sum-Exp function: smooth approximation to maximum.

- Next time:
  - What if we don't know which features are relevant?