CPSC 340: Machine Learning and Data Mining

Gradient Descent

Admin

- Assignment 2:
  - Today is the last day to turn it in (with all late days).
  - We will release solutions tomorrow.
  - We won't make them available to other students
    - unless I hear from the person who wants to opt out
- Assignment 3:
  - Due in 11 days
- Volunteers wanted: marking for Math Challengers competition
  - This is a grade 8/9 math competition that I help organize (picking the questions)
  - Saturday Feb 18th 9:30am-2pm, at UBC
  - If interested, email mathchallengersvolunteers@gmail.com and CC me
  - (there are no incentives or anything, just if you actually want to do it)

Admin (continued)

- Grading disputes
  - If you perceive a problem with your homework grade, you have 1 week to contact us via GitHub
  - Make sure to tag @cpsc340/staff or the particular graders listen on Piazza
  - Since this is a new policy, you have 1 week from today for hw1
  - More detailed info added to General Homework Instructions
- hw3 partnerships
  - Sorry, I'm going to stick to my guns here
  - If you're worried about this happening again, you can open all future issues now (for hw4, hw5, hw6)
  - You can still of course discuss the assignment, but will need to write separate solutions and code
  - I feel bad about this but need to stand up for myself at some point

Last Time: RBFs and Regularization

- We discussed radial basis functions:
  - Basis functions that depend on distances to training points:

$$y_i = w_1 \exp\left(-\frac{\|x_i - x_1\|^2}{2\sigma^2}\right) + w_2 \exp\left(-\frac{\|x_i - x_2\|^2}{2\sigma^2}\right) + \cdots + w_n \exp\left(-\frac{\|x_i - x_n\|^2}{2\sigma^2}\right)$$
$$= \sum_{j=1}^{n} w_j \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

  - Flexible bases that can model any continuous function.
- We also discussed regularization:
  - Adding a penalty on the model complexity:

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

  - Best parameter lambda almost always leads to improved test error.
    - L2-regularized least squares is also known as “ridge regression”.

Features with Different Scales

- Consider features with different scales:

Egg (#)	Milk (mL)	Fish (g)	Pasta (cups)
0	250	0	1
1	250	200	1
0	0	0	0.5
2	250	150	0

- Should we convert to some standard 'unit'?
  - It doesn't matter for least squares:
    - w_j*(100 mL) gives the same model as w_j*(0.1 L)
    - w_j will just be 1000 times smaller.
  - It also doesn't matter for decision trees or naïve Bayes.

Features with Different Scales

- Consider features with different scales:

Egg (#)	Milk (mL)	Fish (g)	Pasta (cups)
0	250	0	1
1	250	200	1
0	0	0	0.5
2	250	150	0

- Should we convert to some standard 'unit'?
  - It matters for k-nearest neighbours:
    - KNN will focus on large values more than small values.
  - It matters for regularized least squares:
    - Penalization $|w_j|$ means different things if features 'j' are on different scales.

Standardizing Features

$X = \begin{bmatrix} & 0 & \end{bmatrix}$

average of column 'j'

- It is common to standardize features:
  - For each feature:
    1. Compute mean and standard deviation:

       $$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \qquad \sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\mu_j)^2}$$

    2. Subtract mean and divide by standard deviation:

       Replace x_{ij} with $\dfrac{x_{ij}-\mu_j}{\sigma_j}$

  - Means that change in 'w_j' have similar effect for any feature 'j'.
- Should we regularize the bias?
  - No! The y-intercept can be anywhere, why encourage it to be close to zero?
  - Yes! Regularizing all variables makes solution unique and it easier to compute 'w'.
  - Compromise: regularize the bias by a smaller amount than other variables?
  - I tried digging into the sklearn Ridge Regression code and it looks like "no".

Standardizing Target

- In regression, we sometimes standardize the targets y_i.
  - Puts targets on the same standard scale as standardized features:

$$\text{Replace } y_i \text{ with } \frac{y_i - \mu_y}{\sigma_y}$$

- With standardized target, setting $w = 0$ predicts average y_i:
  - High regularization makes us predict closer to the average value.
- Other common transformations of y_i are logarithm/exponent:

$$\text{Use } \log(y_i) \text{ or } \exp(\gamma y_i)$$

  - Makes sense for geometric/exponential processes.

Ridge Regression Calculation

Objective: $f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2} w^T w$

Gradient: $\nabla f(w) = X^T X w - X^T y + \lambda w$

Set $\nabla f(w) = 0$: $X^T X w + \lambda w = X^T y$

or

$$X^T X w + \lambda I w = X^T y$$

or

$$(X^T X + \lambda I) w = X^T y$$

Pre-multiply by $(X^T X + \lambda I)^{-1}$ which <u>always</u> exists:

$$\underbrace{(X^T X + \lambda I)^{-1}(X^T X + \lambda I)}_{I} w = (X^T X + \lambda I)^{-1} X^T y$$

Python:

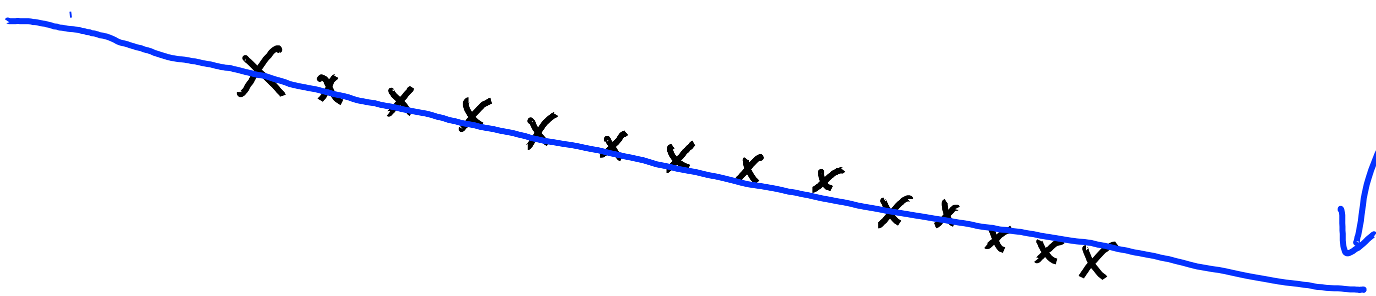
```
w = solve(X.T@X+lam*np.eye(d), X.T*y)
```

so $w = (X^T X + \lambda I)^{-1} X^T y$

Least Squares with Outliers

- Consider least squares problem with outliers:

x ← "outlier" that doesn't follow trend



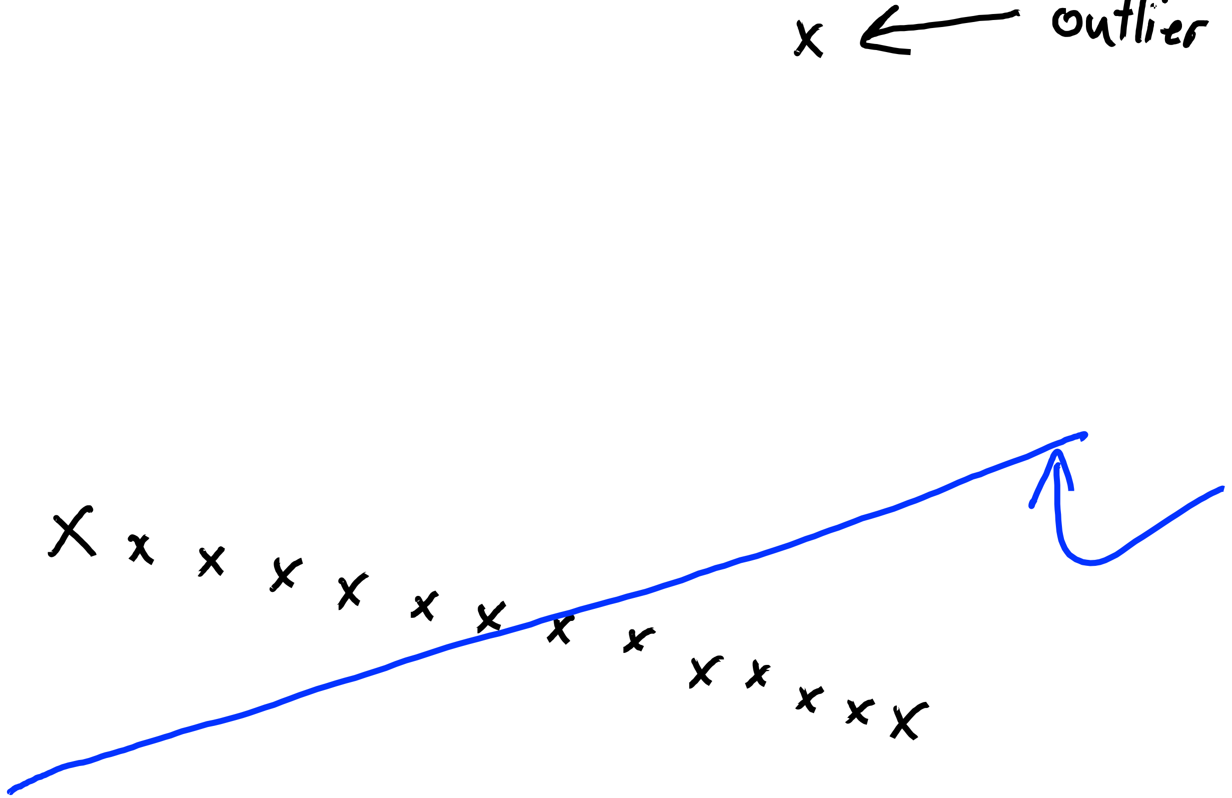
This is what we want least squares to do.

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

Least Squares with Outliers

- Consider least squares problem with outliers:

"outlier" that doesn't follow trend

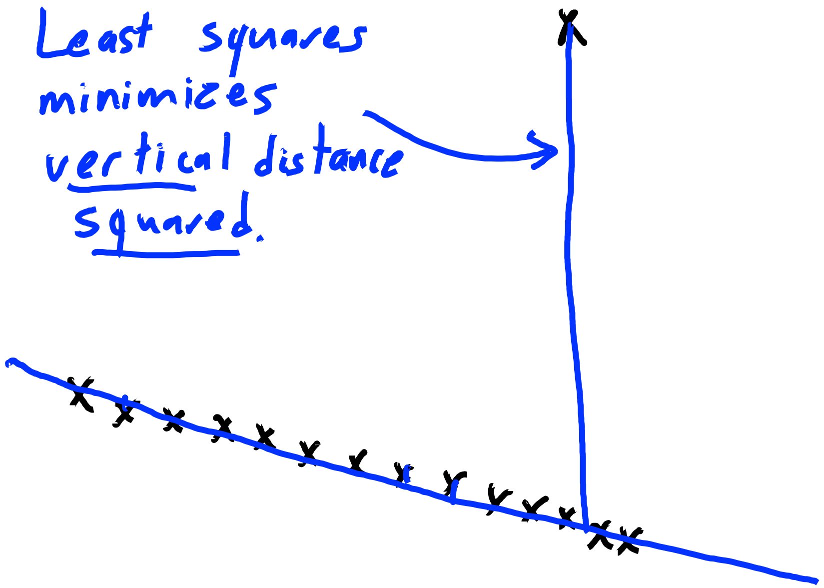


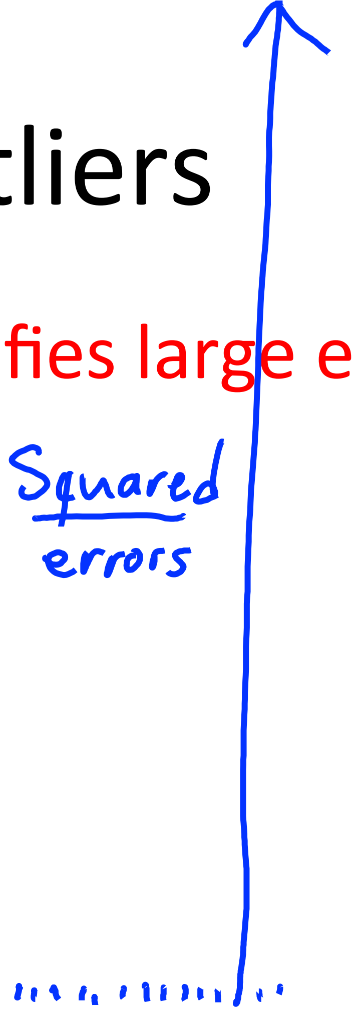
This is what least squares will actually do.

- Least squares is very sensitive to outliers.

Least Squares with Outliers

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

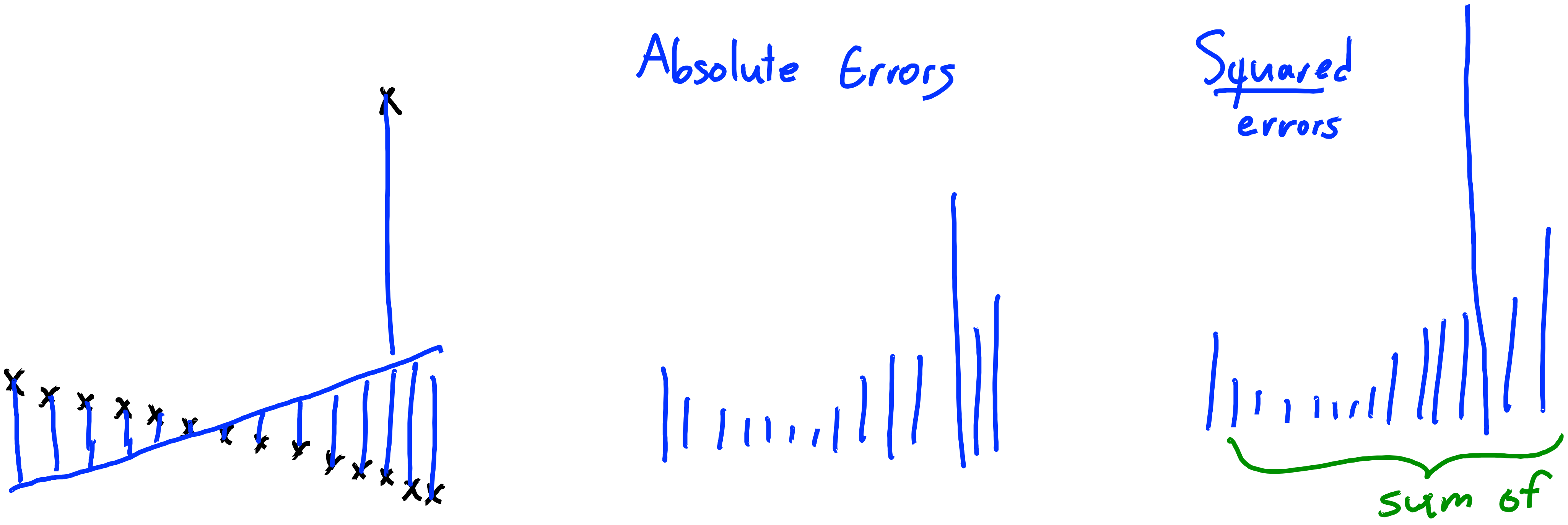




- Outliers (large error) influence ‘w’ much more than other points.

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'.

smaller than for the "correct" line.

Robust Regression

- Robust regression objectives put less focus 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:

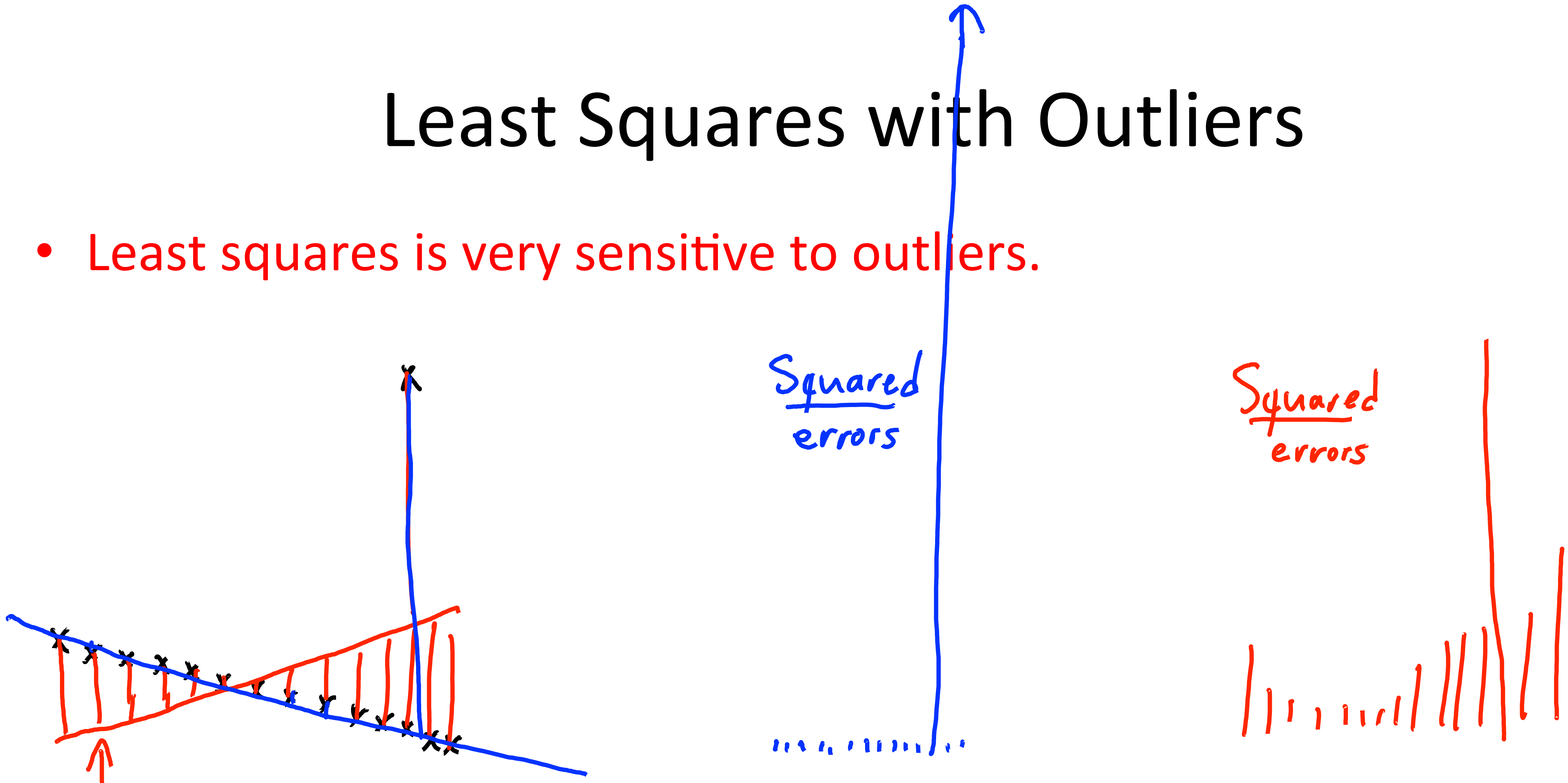
$$f(w) = \frac{1}{2}\|Xw - y\|^2$$

Least absolute error:

$$f(w) = \|\underbrace{X}_{n \times d}\underbrace{w}_{d \times 1} - \underbrace{y}_{n \times 1}\|_1$$

Least Squares with Outliers

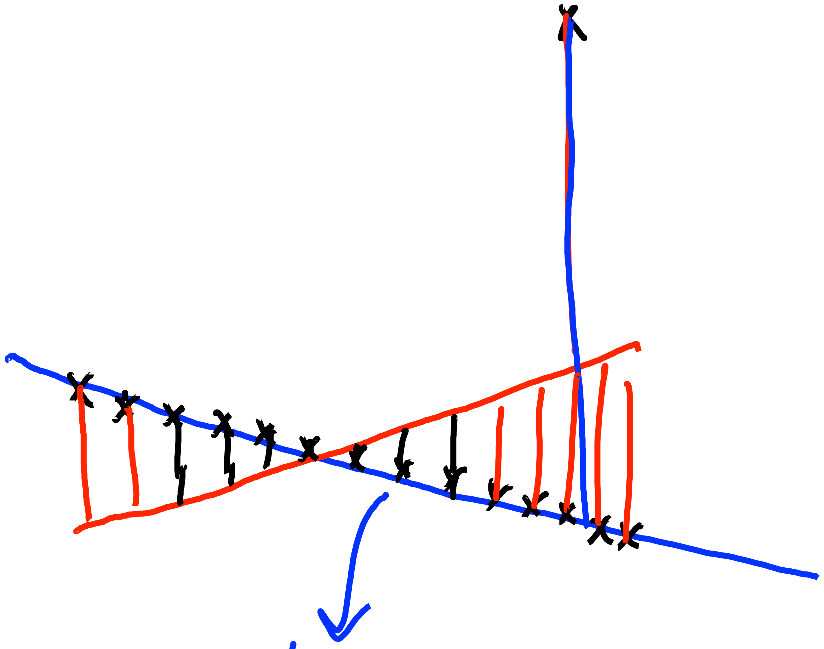
- Least squares is very sensitive to outliers.



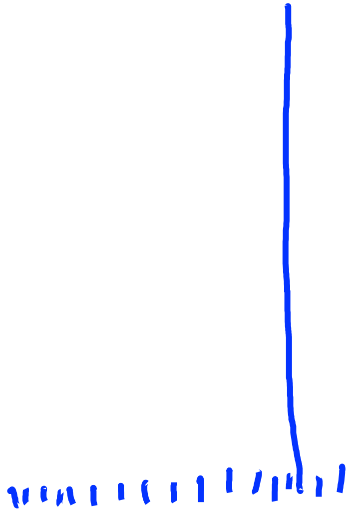
Linear model 'w' minimizing $f(w) = \frac{1}{2}\|Xw - y\|^2$

Least Squares with Outliers

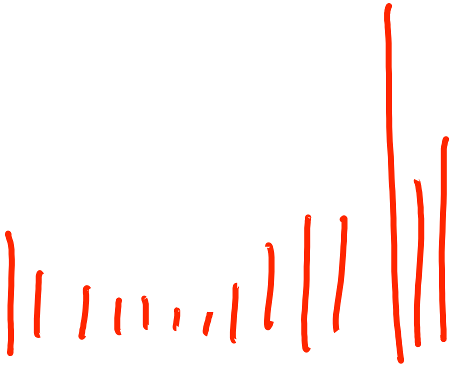
- Absolute error is more robust to outliers:



Absolute Errors



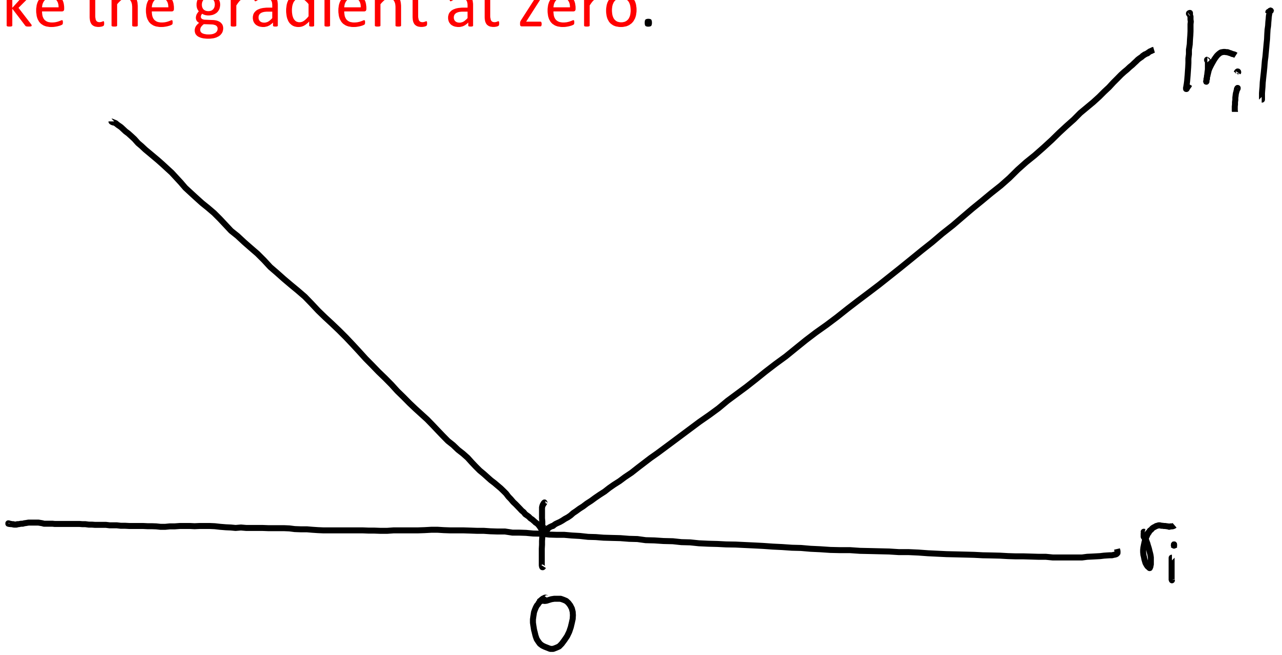
Absolute Errors



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 can't take the gradient at zero.



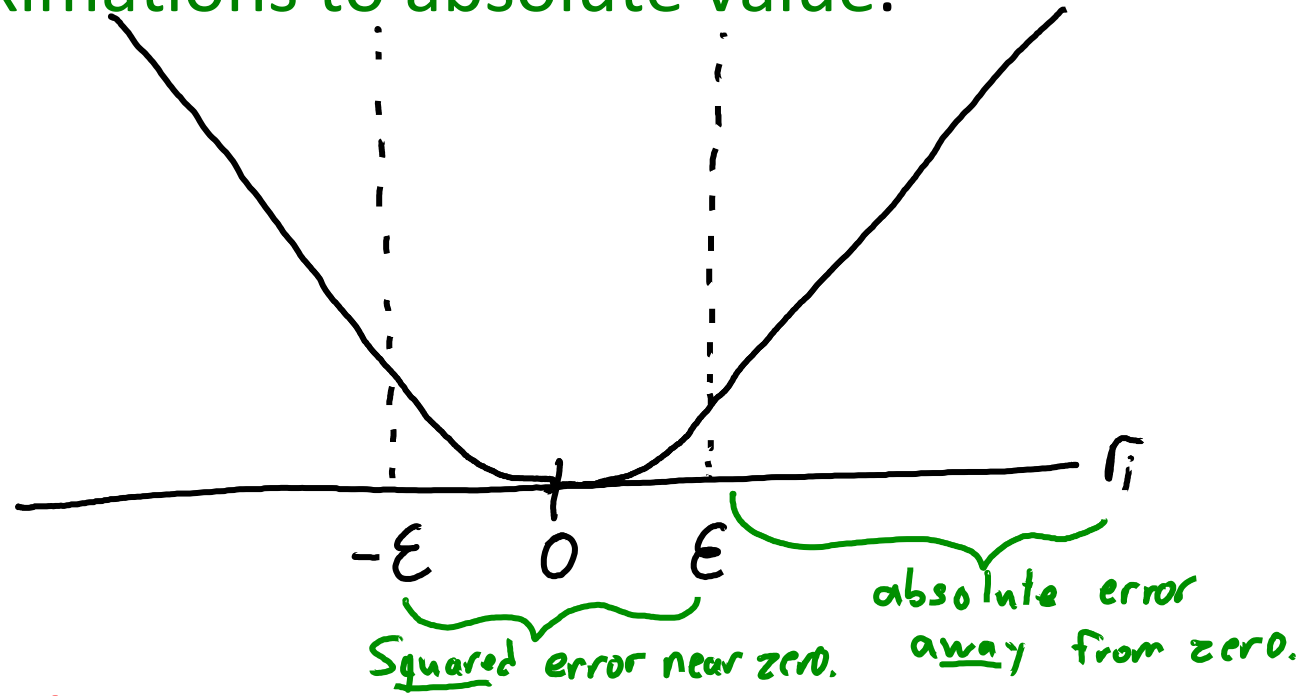
  - Generally, harder to minimize non-smooth than smooth functions.
  - Could solve as 'linear program', but harder than 'linear system'.

Smooth Approximations to the L1-Norm

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

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

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



- Setting $\nabla f(x) = 0$ does not give a linear system.
- But we can minimize 'f' using gradient descent:
  - Algorithm for finding local minimum of a differentiable function.

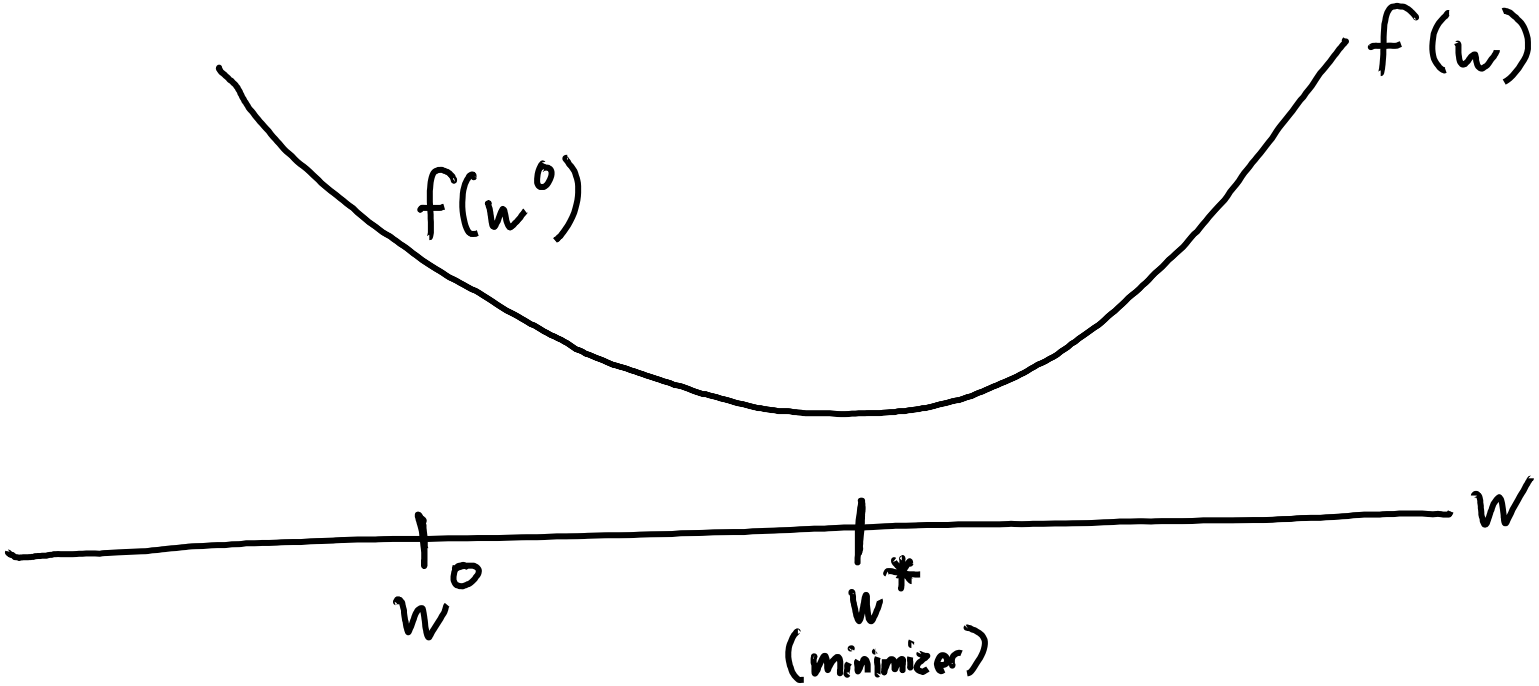
Gradient Descent for Finding a Local Minimum

- Gradient descent is an iterative optimization algorithm:
  - It starts with a "guess" w^0.
  - It uses w^0 to generate a better guess w^1.
  - It uses w^1 to generate a better guess w^2.
  - It uses w^2 to generate a better guess w^3.

    …
  - The limit of w^t as 't' goes to ∞ has $\nabla f(w^t) = 0$.

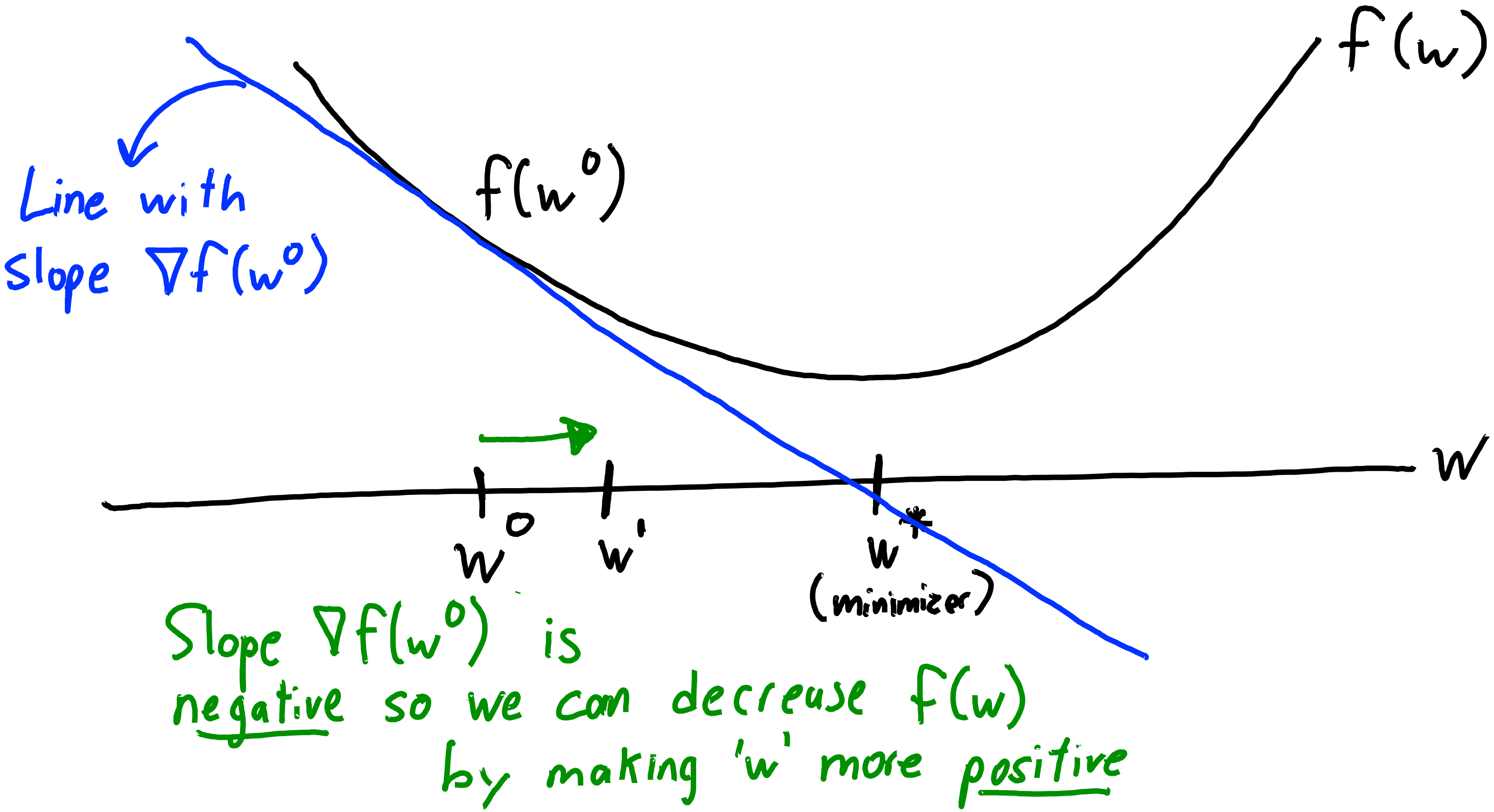
Gradient Descent for Finding a Local Minimum

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



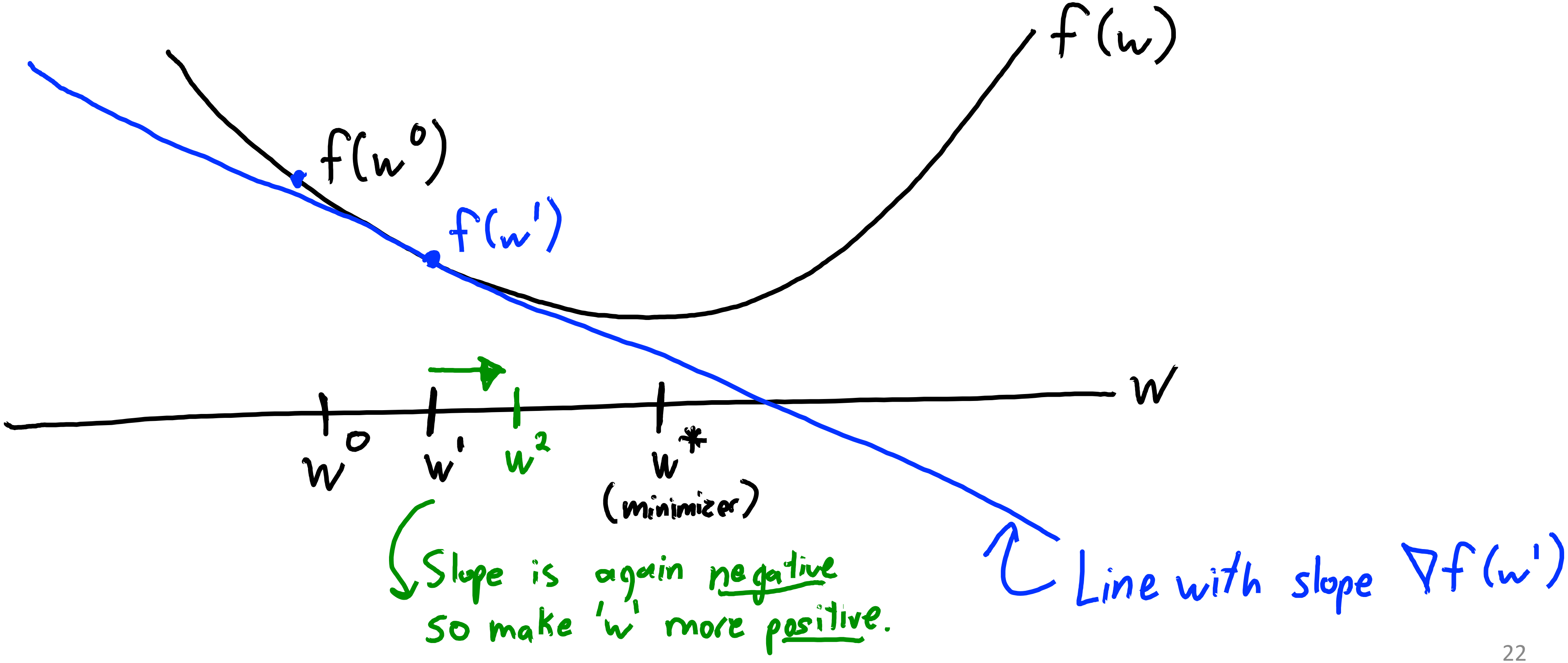
Gradient Descent for Finding a Local Minimum

- Gradient descent is based on a simple observation:
  - Give parameters 'w', the direction of largest decrease is $-\nabla f(w)$.



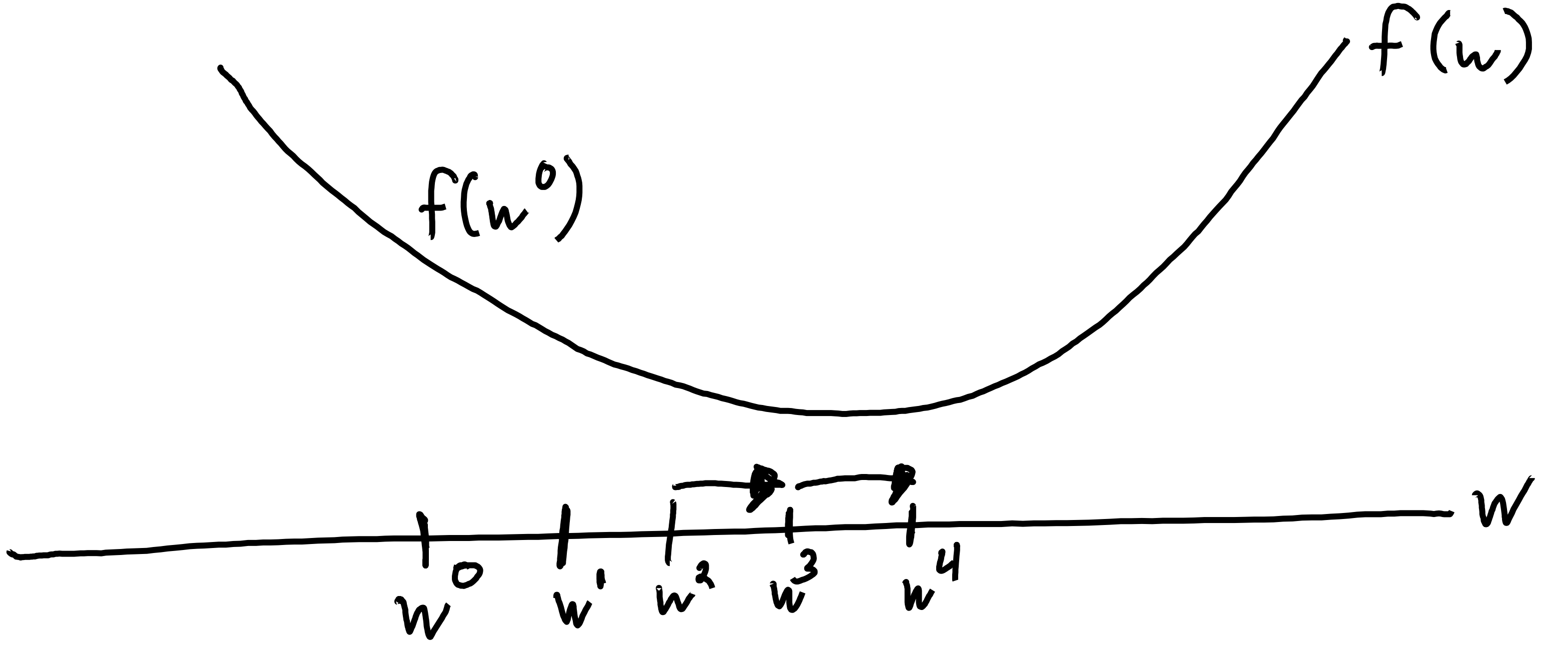
Gradient Descent for Finding a Local Minimum

- Gradient descent is based on a simple observation:
  - Give parameters 'w', the direction of largest decrease is $-\nabla f(w)$.



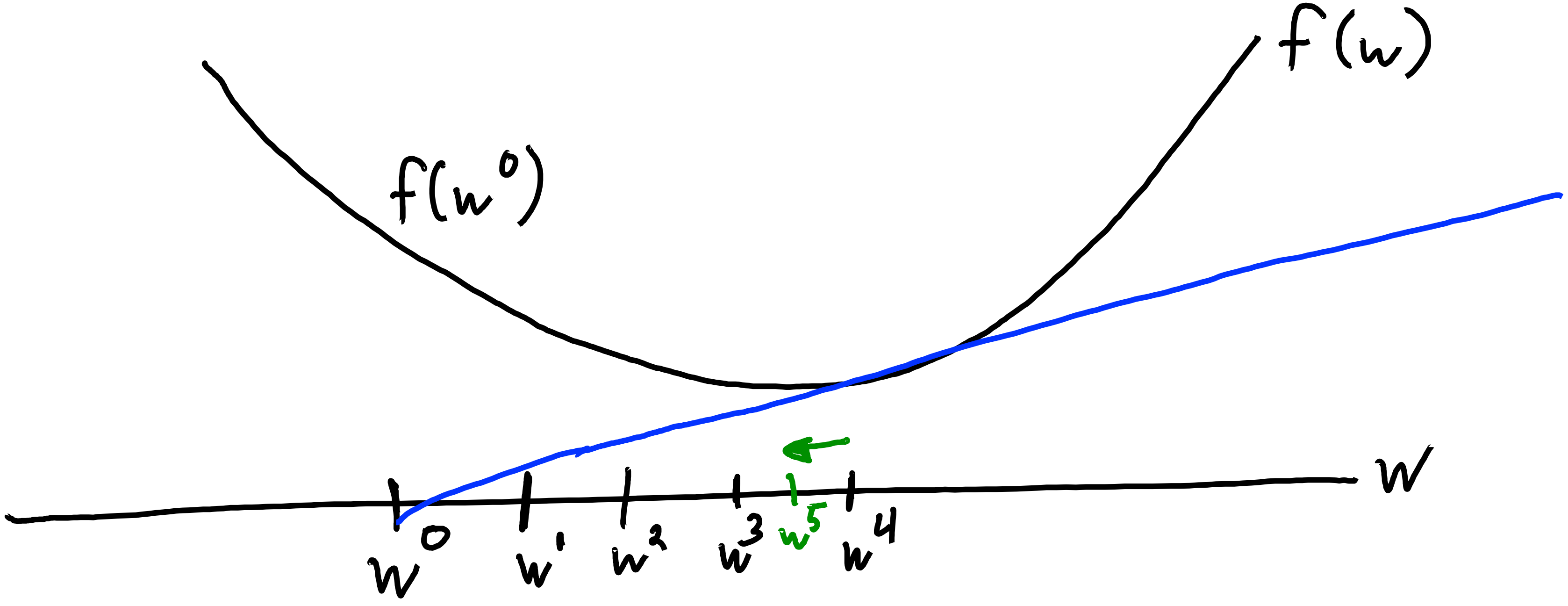
Gradient Descent for Finding a Local Minimum

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



Gradient Descent for Finding a Local Minimum

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



Now the slope $\nabla f(w^4)$ is positive
so we move in the negative direction.

Gradient Descent for Finding a Local Minimum

- Gradient descent is an iterative optimization algorithm:
  - It starts with a "guess" w^0.
  - Generate new guess by moving in the negative gradient direction:

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

  - This decreases f if the step size α is small enough
  - Repeat to successively refine the guess

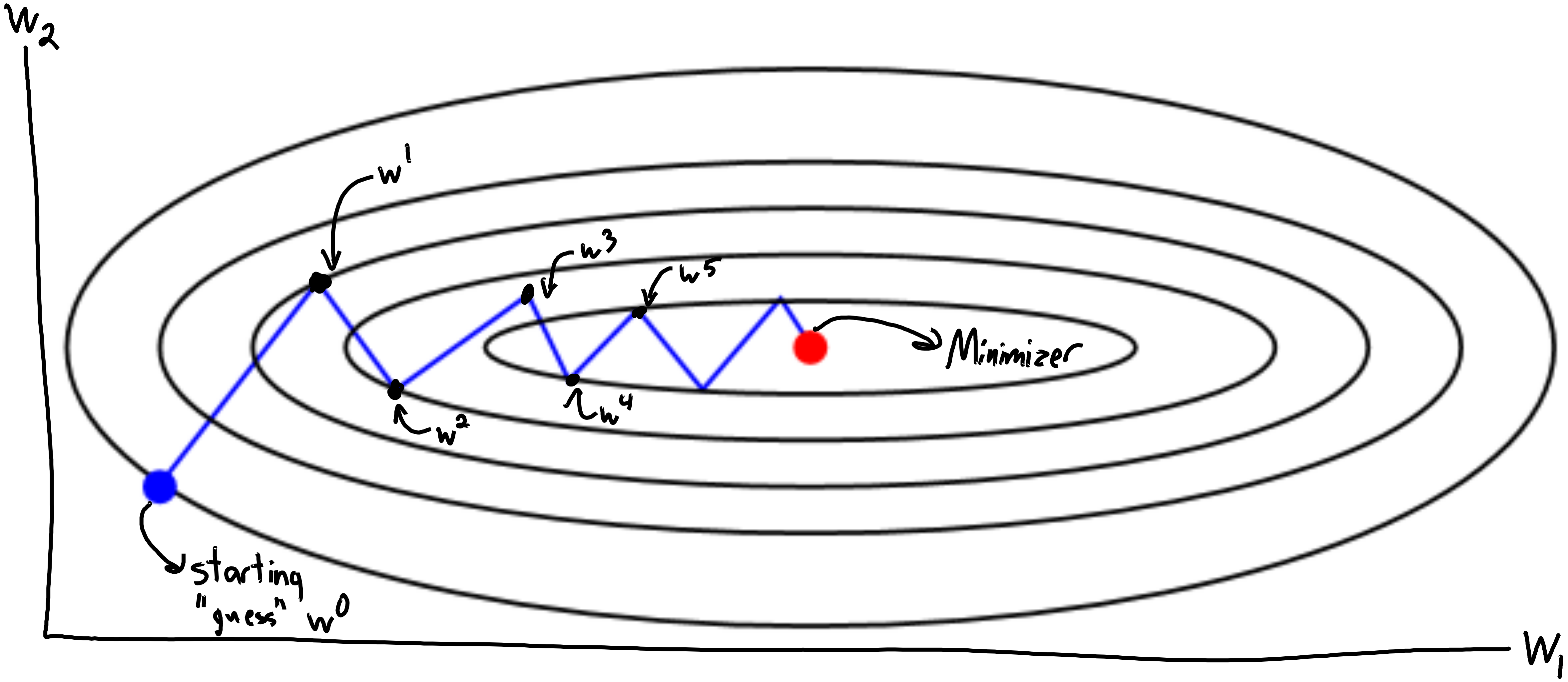
$$w^{t+1} = w^t - \alpha^t \nabla f(w^t) \quad \text{for } t = 1, 2, 3, \ldots$$

  - Stop if not making progress or

$$\underbrace{\|\nabla f(w^t)\| \leq \delta}_{\text{Approximate local minimum}}$$

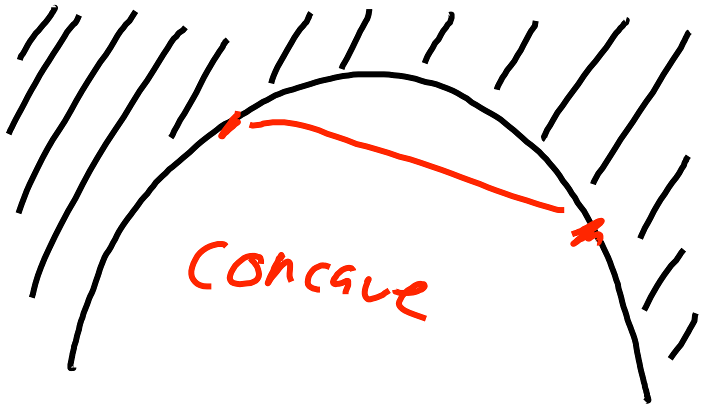
$\delta \rightarrow$ Some small scalar.

Gradient Descent in 2D

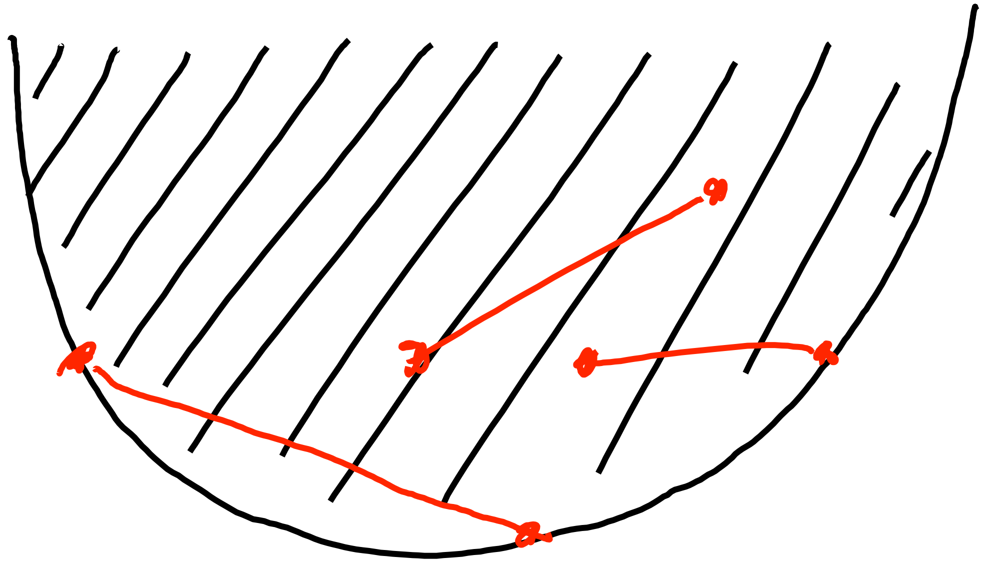


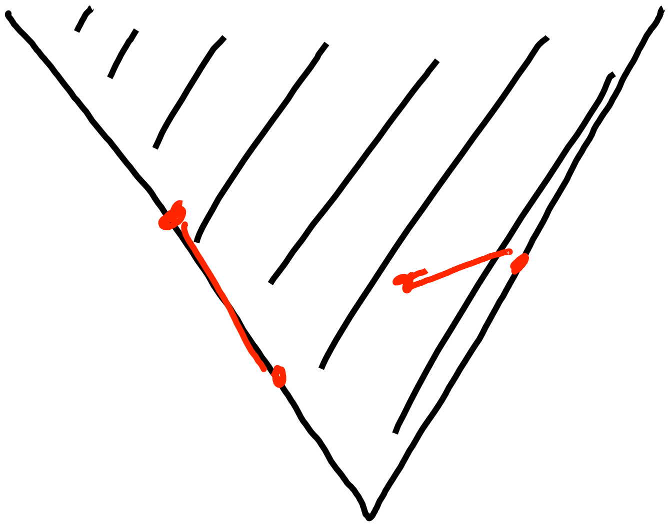
- Under weak conditions, algorithm converges to a local minimum.

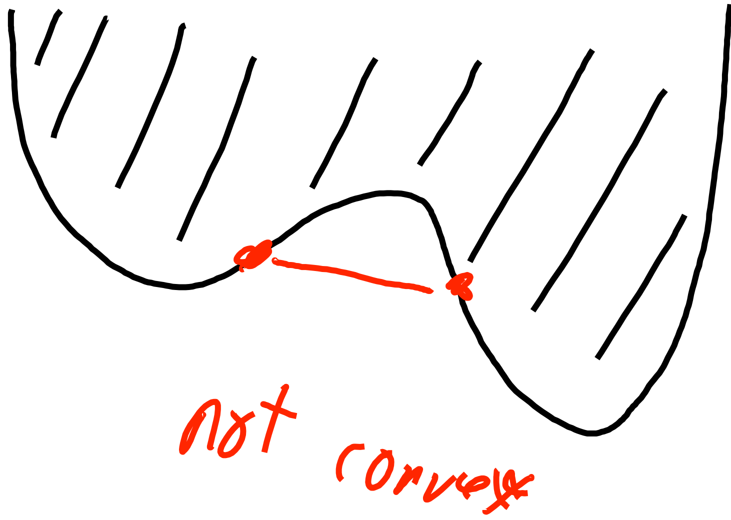
Convex Functions



- Is finding a local minimum good enough?
  - For least squares and Huber loss this is enough: they are convex functions.







- A function is convex if the area above the function is a convex set.
  - All values between any two points above function stay above function.

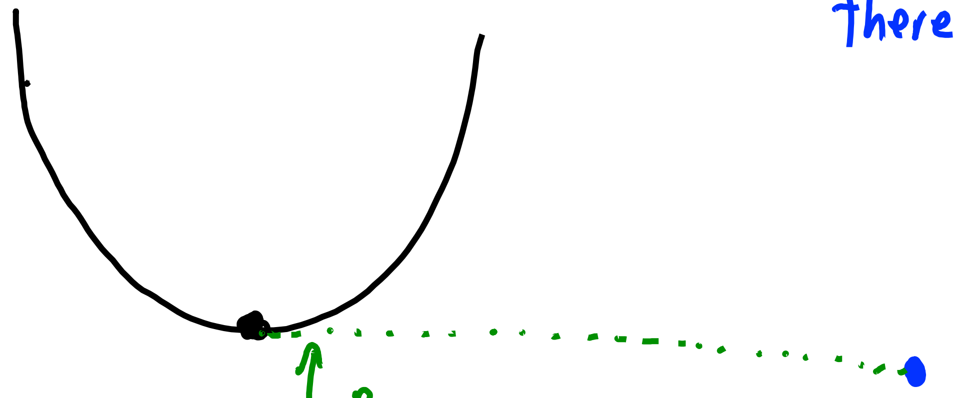
Convex Functions

- All local minima of convex functions are also global minima.

Proof by contradiction:

Consider a local minimum

If this is not global minimum, there must a smaller value.



But this contradicts that we are at a local minimum.

By convexity we can move along line to global minimum and decrease objective.

  - Gradient descent finds a global minimum on convex functions.
  - Next time: how do we know if a function is convex?

Gradient Descent

- Least squares via normal equations vs. gradient descent:
  - Normal equations cost $O(nd^2 + d^3)$.

    Forming X^TX costs $O(nd^2)$ and solving a $d \times d$ linear system costs $O(d^3)$

  - Gradient descent costs $O(ndt)$ to run for 't' iterations.

    Computing $\nabla f(w) = X^TXw - X^Ty$ only costs $O(nd)$.

    We can use $X^TXw = X^T(Xw)$ which is just two $n \times d$ matrix multiplications.

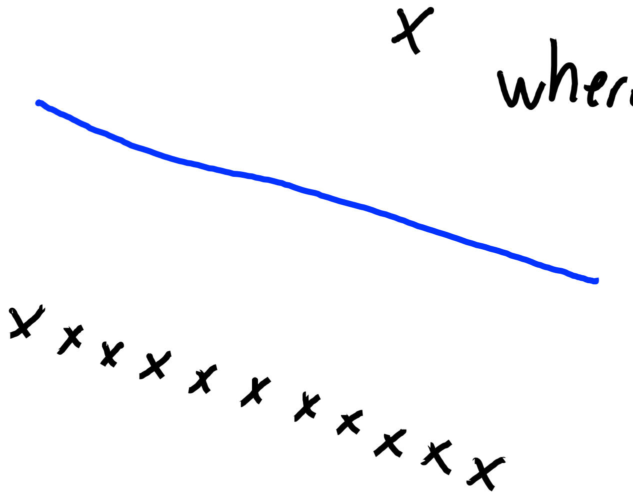
  - Gradient descent can be faster when 'd' is very large:
- Improving on gradient descent: Nesterov and Newton method.
  - For L2-regularized least squares, there is also "conjugate" gradient.

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:

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



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

Log-Sum-Exp Function

- As with the L_1-norm, the L_∞ norm is convex but non-smooth:
  - We can fit it with gradient descent using a smooth approximation.
- Log-sum-exp function is a smooth approximation to the max function:

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

- Intuition: largest element is magnified exponentially.
  - Smaller elements become negligible in comparison.
  - Recall that log(exp(z))=z.
- Notation alert: by "log" I always mean the natural logarithm.

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:
  - Finding 'important' e-mails, and beating naïve Bayes on spam filtering.