Lecture 5 – Gradient Descent and Convexity

DSC 40A, Fall 2022 @ UC San Diego

Mahdi Soleymani, with help from **many others**

Agenda

- Minimizing UCSD loss.
- Gradient descent fundamentals.

A new loss function

The recipe

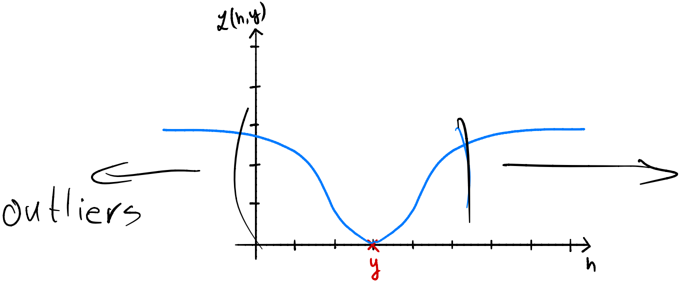
Suppose we're given a dataset, $y_1, y_2, ..., y_n$ and want to determine the best future prediction h^*.
The recipe is as follows:

1. Choose a loss function $L(h, y)$ that measures how far our prediction h is from the "right answer" y.
   - Absolute loss, $L_{abs}(h, y) = |y - h|$.
   - Squared loss, $L_{sq}(h, y) = (y - h)^2$.
2. Find h^* by minimizing the average of our chosen loss function over the entire dataset.
   - "Empirical risk" is just another name for average loss.

$$R(h) = \frac{1}{n} \sum_{i=1}^{n} L(h, y)$$

A very insensitive loss

- Last time, we introduced a new loss function, L_{ucsd}, with the property that it (roughly) penalizes all bad predictions the same.
  - Under L_{ucsd}, a prediction that is wrong by 50 has approximately the same loss as a prediction that is wrong by 500.
  - The effect: L_{ucsd} is not as sensitive to outliers.



Adding a scale parameter

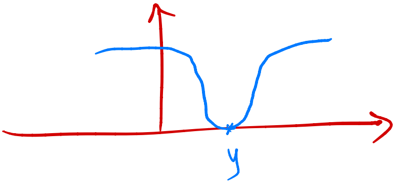
- Problem: L_{ucsd} has a fixed scale. This won't work for all datasets.
  - If we're predicting temperature, and we're off by 100 degrees, that's bad.
  - If we're predicting salaries, and we're off by 100 dollars, that's pretty good.
  - What we consider to be an outlier depends on the scale of the data.
- Fix: add a **scale parameter**, σ:

scaler

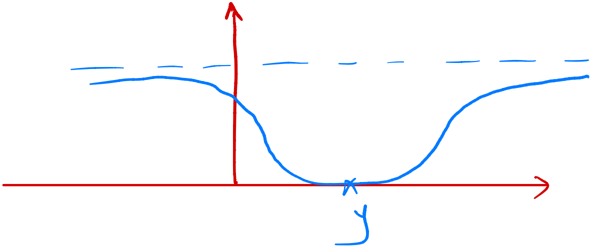
$$L_{ucsd}(h, y) = 1 - e^{-(y-h)^2/\sigma^2}$$

Adding a scale parameter

$$L_{ucsd} = 1 - e^{\left[-\frac{(y-h)^2}{\sigma^2}\right]}$$



small σ



Large σ

Empirical risk minimization

- We have salaries $y_1, y_2, ..., y_n$.
- To find prediction, ERM says to minimize the average loss:

$$\underbrace{R_{ucsd}}(h) = \frac{1}{n}\sum_{i=1}^{n} L_{ucsd}(h, y_i)$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[1 - e^{-(y_i-h)^2/\sigma^2}\right]$$

σ^2 is not the variance.

Let's plot R_{ucsd}

h^*

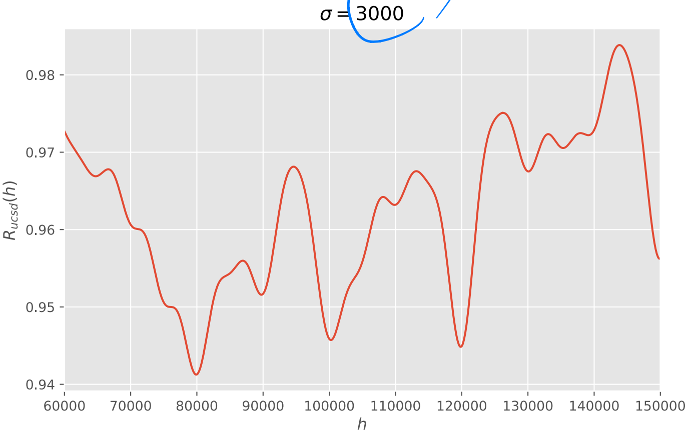
- Recall:

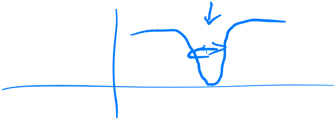
$$R_{ucsd}(h) = \frac{1}{n}\sum_{i=1}^{n}\left[1 - e^{-(y_i-h)^2/\sigma^2}\right]$$

- Once we have data $y_1, y_2, \ldots, y_n$ and a scale σ, we can plot $R_{ucsd}(h)$.
- We'll use full the StackOverflow dataset ($n = 1121$).
- Let's try several scales, σ.

Plot of $R_{ucsd}(h)$

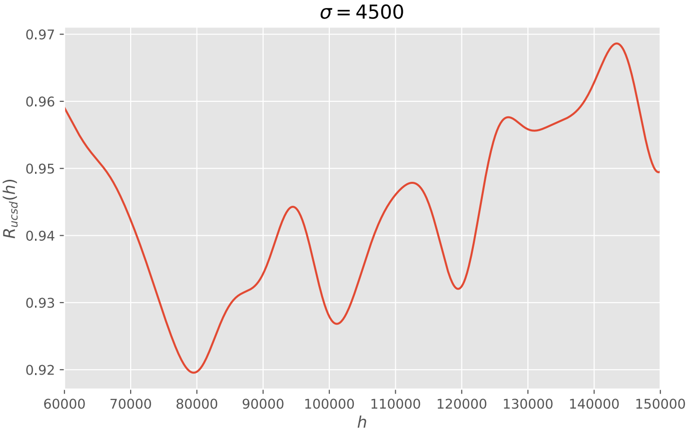
salaries 10^4 - 10^5



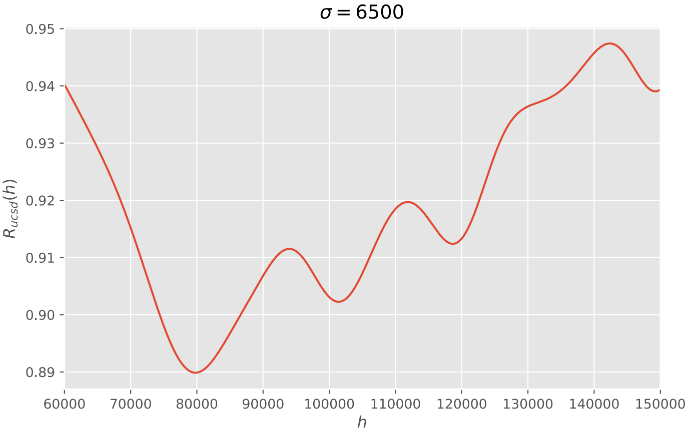


larger σ → wider the loss function

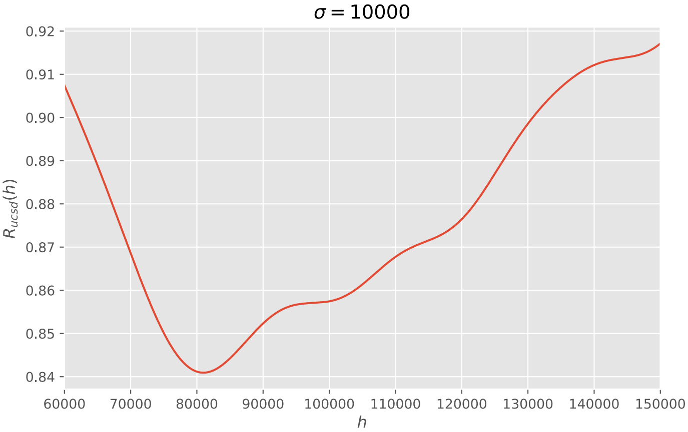
Plot of $R_{ucsd}(h)$



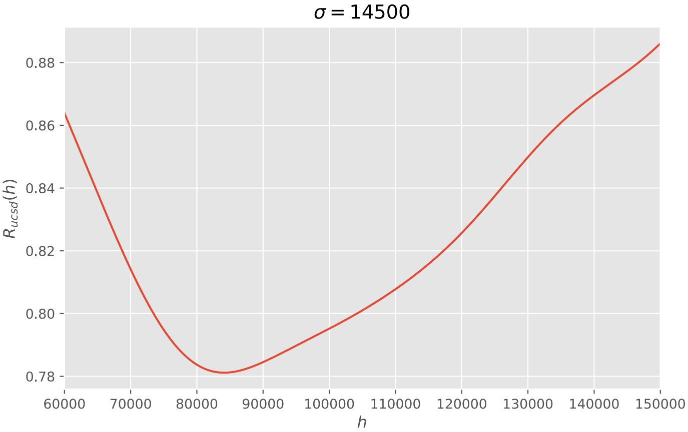
Plot of $R_{ucsd}(h)$



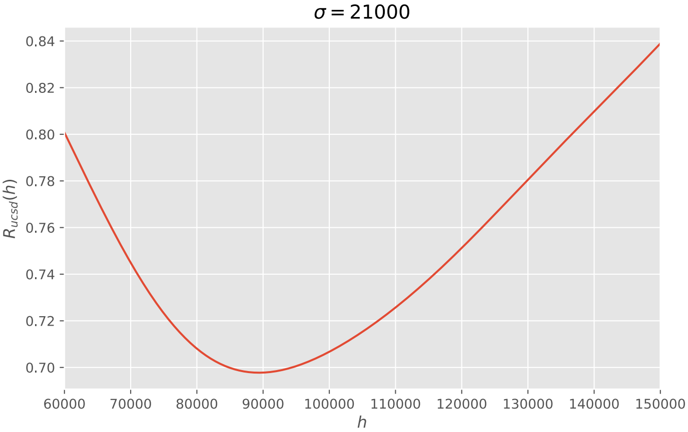
Plot of $R_{ucsd}(h)$



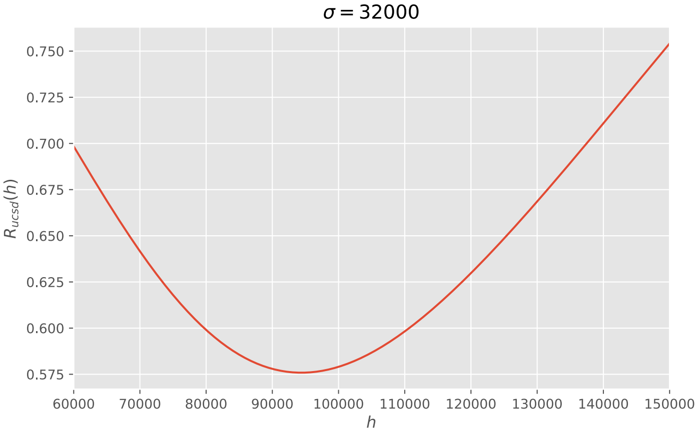
Plot of $R_{ucsd}(h)$



Plot of $R_{ucsd}(h)$



Plot of $R_{ucsd}(h)$



Minimizing R_{ucsd}

- To find the best prediction, we find h^* minimizing $R_{ucsd}(h)$.
- $R_{ucsd}(h)$ is **differentiable**.
- To minimize: take derivative, set to zero, solve.

$$R_{UCSD} = \frac{1}{n} \sum_{i=1}^{n} \left[1 - e^{-\frac{(y_i - h)^2}{\sigma^2}} \right]$$

Step 1: Taking the derivative

$$\frac{dR_{ucsd}}{dh} = \frac{d}{dh}\left(\frac{1}{n}\sum_{i=1}^{n}\left[1 - e^{-(y_i-h)^2/\sigma^2}\right]\right)$$

$$\frac{d}{dh}\ldots$$

$$= \frac{1}{n}\left[\sum_{i=1}^{n} \frac{d}{dh} 1 - \sum_{i=1}^{n} \frac{d}{dh} e^{-(y_i-h)^2/\sigma^2}\right]$$

$$= \frac{1}{n}\left[\sum_{i=1}^{n} 0 - \sum_{i=1}^{n} e^{-\frac{(y_i-h)^2}{\sigma^2}} \times \frac{-2(y_i-h)}{\sigma^2} \times -1\right]$$

$$= \frac{2}{n\sigma^2}\sum_{i=1}^{n} e^{-\frac{(y_i-h)^2}{\sigma^2}}(h-y_i)$$

Step 2: Setting to zero and solving

h^*

- We found:

$$\frac{d}{dh}(h) = \frac{2}{n\sigma^2} \sum_{i=1}^{n} (h - y_i) \cdot e^{-(h-y_i)^2/\sigma^2} = 0$$

- Now we just set to zero and solve for h:

$ax^2 + bx + c = 0$

$\Rightarrow x^* = \frac{-b \pm \sqrt{b^2 - 4ac}}{2}$

$$0 = \frac{2}{n\sigma^2} \sum_{i=1}^{n} (h - y_i) \cdot e^{-(h-y_i)^2/\sigma^2}$$

- We **can** calculate derivative, but we **can't** solve for h; we're stuck again.

- Now what???

L_{ucsd}

- The formula for L_{ucsd} is as follows (no need to memorize):

$$L_{ucsd}(h, y) = 1 - e^{-(y-h)^2/\sigma^2}$$

  - The shape (and formula) come from an upside-down bell curve.

- L_{ucsd} contains a **scale parameter**, σ.
  - Nothing to do with variance or standard deviation.
  - Accounts for the fact that different datasets have different thresholds for what counts as an outlier.
  - Think of σ as a knob that you get to turn – the larger σ is, the more sensitive L_{ucsd} is to outliers (and the more smooth R_{ucsd} is).

There's a problem with R_{ucsd}

- The corresponding empirical risk, R_{ucsd}, is

$$R_{ucsd}(h) = \frac{1}{n} \sum_{i=1}^{n} \left[1 - e^{-(y_i - h)^2 / \sigma^2}\right]$$

- R_{ucsd} is **differentiable**.
- Last time, we took the derivative of $R_{ucsd}(h)$ and set it equal to 0.

$$0 = \frac{2}{n\sigma^2} \sum_{i=1}^{n} (h - y_i) \cdot e^{-(y_i - h)^2 / \sigma^2}$$

- There's no solution to this equation. So now what?

Gradient descent fundamentals

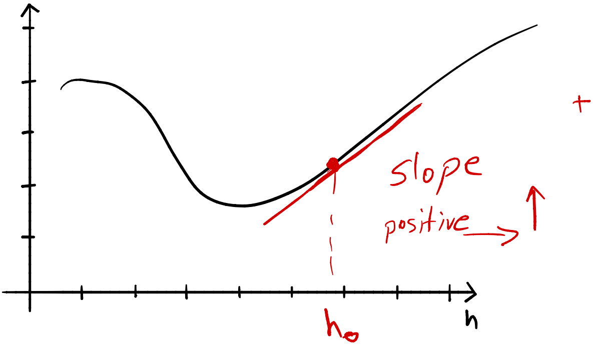
The general problem

- **Given:** a differentiable function $R(h)$.

- **Goal:** find the input h^* that minimizes $R(h)$.

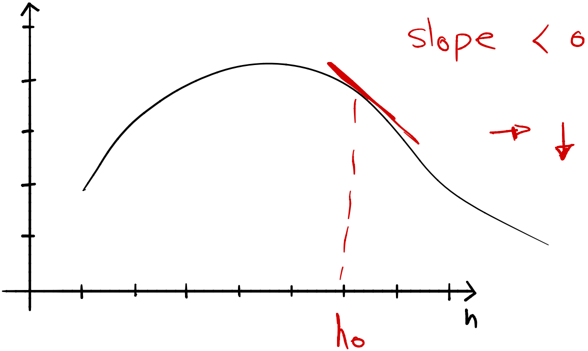
Meaning of the derivative

- We're trying to minimize a **differentiable** function $R(h)$. Is calculating the derivative helpful?
- $\frac{dR}{dh}(h)$ is a function; it gives the **slope** at h.



Key idea behind gradient descent

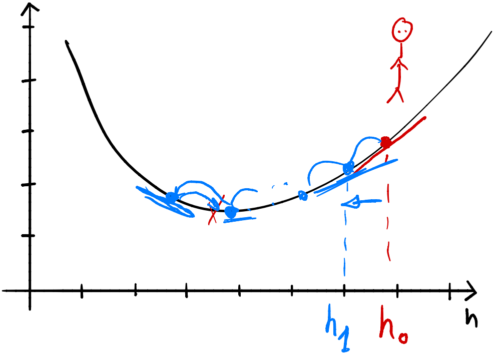
- If the slope of R at h is **positive** then moving to the **left** decreases the value of R.
- i.e., we should **decrease** h.



Key idea behind gradient descent

- If the slope of R at h is **negative** then moving to the **right** decreases the value of R.
- i.e., we should **increase** h.

$R(h) =$ Given



Key idea behind gradient descent

$R(h)$

- Pick a starting place, h_0. Where do we go next?
- Slope at h_0 negative? Then increase h_0.
- Slope at h_0 positive? Then decrease h_0.
- This will work:

$$\boxed{h_1 = h_0 - \frac{dR}{dh}(h_0)}$$

update equation

Gradient Descent

- Pick α to be a positive number. It is the **learning rate**, also known as the **step size**.

random point

- Pick a starting prediction, h_0.

step size

maximum

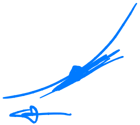
+

- On step i, perform update $h_i = h_{i-1} - \alpha \cdot \frac{dR}{dh}(h_{i-1})$

positive

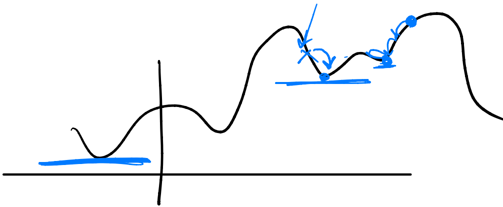
- Repeat until convergence (when h doesn't change much).

- **Note:** it's called gradient descent because the "gradient" is the generalization of the derivative for multivariate functions.



You will not be responsible for implementing gradient descent in this class, but here's an implementation in Python if you're curious:

```
def gradient_descent(derivative, h, alpha, tol=1e-12):
    """Minimize using gradient descent."""
    while True:
        h_next = h - alpha * derivative(h)
        if abs(h_next - h) < tol:
            break
        h = h_next
    return h
```



Example: Minimizing mean squared error

- Recall the mean squared error and its derivative:

$$R_{sq}(h) = \frac{1}{n}\sum_{i=1}^{n}(y_i - h)^2 \qquad \frac{dR_{sq}}{dh}(h) = \frac{2}{n}\sum_{i=1}^{n}(h - y_i)$$

$h_i = h_{i-1} - \alpha \times \frac{dR}{dh}(h_0)$

Discussion Question

Let $y_1 = -4$, $y_2 = -2$, $y_3 = 2$, $y_4 = 4$. Pick $h_0 = 4$ and $\alpha = 1/4$. What is h_1?

$h_1 = h_0 - \alpha \frac{dR}{dh}(h_0)$

a) -1
b) 0
c) 1
d) 2

To answer, go to `menti.com` **and enter the code 7933 4859.**

Solution

$$R_{sq}(h) = \frac{1}{n}\sum_{i=1}^{n}(y_i - h)^2 \qquad \frac{dR_{sq}}{dh}(h) = \frac{2}{n}\sum_{i=1}^{n}(h - y_i)$$

Data values are $-4, -2, 2, 4$. Pick $h_0 = 4$ and $\alpha = 1/4$. Find h_1.

$$h_i = h_{i-1} - \alpha \frac{dR_{sq}}{dh}(h_{i-1})$$

$$\Rightarrow h_1 = h_0 - \alpha \frac{dR}{dh}(h_0)$$

$$\Rightarrow h_1 = 4 - \frac{1}{4} \times 8 = 2$$

$$\frac{dR(4)}{dh} = \frac{2}{4}\left[(4-(-4)) + (4-(-2)) + (4-2) + (4-4)\right] = \frac{1}{2}\cdot 16 = \boxed{8}$$