Lecture 5 – Gradient Descent and Convexity

DSC 40A, Fall 2021 @ UC San Diego

Suraj Rampure, with help from **many others**

Announcements

- Supplemental videos from last quarter are posted at `dsc40a.com/resources`.
- Groupwork 2 is due **tonight at 11:59pm**.
- Homework 2 is due **on Monday at 11:59pm**.
  - Survey 2 will come out soon; fill it out after Homework 2.
- Make sure to fill out Survey 1!
- Homework 1 solutions are available on Campuswire.

→ Course expanded to 146 (6 more seats).

Agenda

- Brief recap of Lecture 4.
- Gradient descent fundamentals.
- Gradient descent demo.
- When is gradient descent guaranteed to work?
    - Will discuss the idea of “convexity”.

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:

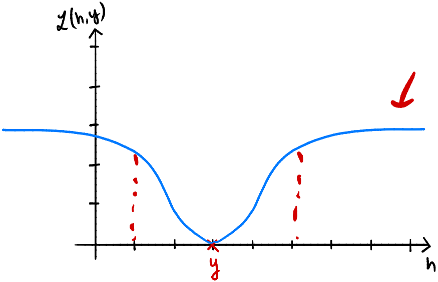
1. Choose a loss function $L(h, y)$ that measures how far our prediction h is from the "right answer" y.
    - Absolute loss, $L_{abs}(h, y) = |y - h|$.
    - Squared loss, $L_{sq}(h, y) = (y - h)^2$.
2. Find h^* by minimizing the average of our chosen loss function over the entire dataset.
    - "Empirical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

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

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.



loss is flat

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.

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

can't solve by hand

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

Gradient descent fundamentals

The general problem

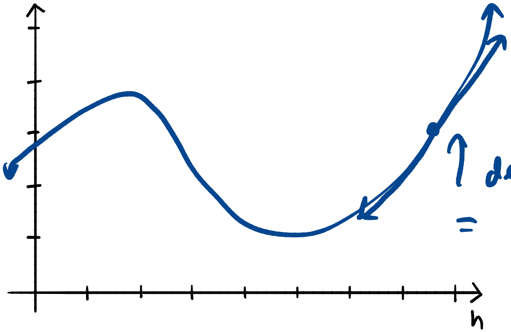
- **Given:** a differentiable function $R(h)$.

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

trying to find best prediction

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.

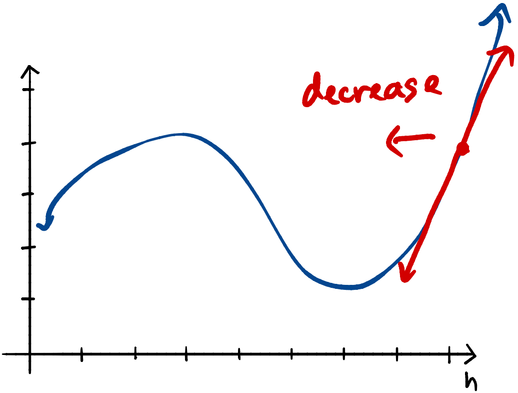


↑ derivative = slope of tangent

Key idea behind gradient descent

- If the slope of R at h is **positive** then moving to the **left** decreases the value of R.

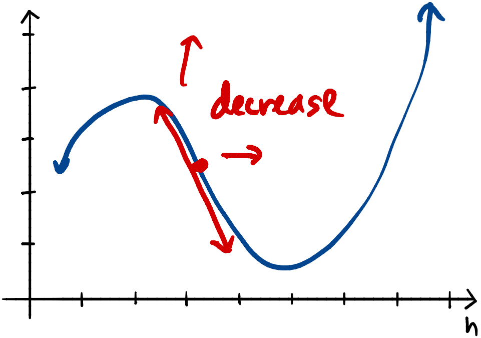
- i.e., we should **decrease** h.



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.



Key idea behind gradient descent

- 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:

$$h_1 = \overset{\text{initial guess}}{h_0} \underbrace{- \frac{dR}{dh}(h_0)}_{\text{opposite the direction of the derivative}}$$

Gradient Descent

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

- Pick a starting prediction, h_0. → "h naught"

- On step i, perform update $h_i = h_{i-1} - \overset{\text{learning rate/step size}}{\alpha} \cdot \underbrace{\frac{dR}{dh}(h_{i-1})}_{\text{derivative at prev. guess}}$

- 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:

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

Discussion Question

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

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

To answer, go to `menti.com` **and enter the code 7910 4287.**

Solution

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

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

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

$$\Rightarrow h_1 = \overset{4}{h_0} - \overset{\frac{1}{4}}{\alpha} \cdot \frac{dR_{sq}}{dh}\overset{8}{(h_0)} = 4 - \frac{1}{4}\cdot 8 = 4 - 2 = \overset{h_1}{\boxed{2}}$$

$$\Rightarrow \frac{dR_{sq}}{dh}(4) = \frac{2}{4}\Big[[4-(-4)] + [4-(-2)] + [4-2] + [4-4]\Big]$$

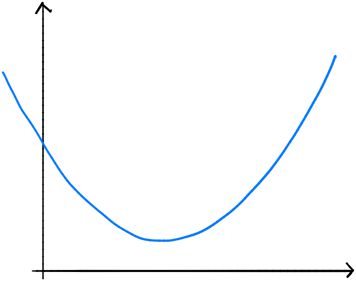
$$= \frac{1}{2}\cdot 16 = \boxed{8}$$

Gradient descent demo

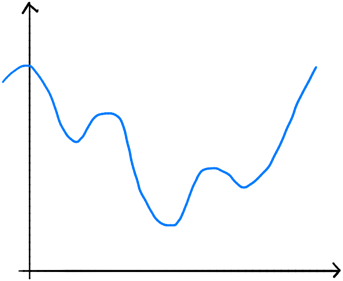
Let’s see gradient descent in action.
Follow along with the demo by clicking the **code** link on the course website next to Lecture 5.

When is gradient descent guaranteed to work?

Convex functions



Convex



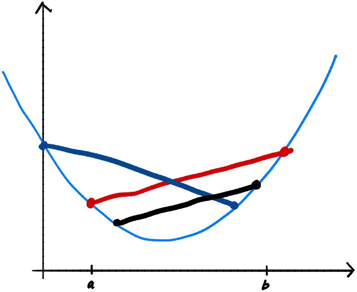
Non-convex

Convexity: Definition

- f is **convex** if for **every** a, b in the domain of f, the line segment between

$$(a, f(a)) \qquad \text{and} \qquad (b, f(b))$$

does not go below the plot of f.

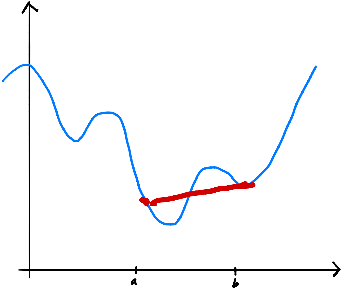


Convexity: Definition

- f is **convex** if for **every** a, b in the domain of f, the line segment between

$$(a, f(a)) \qquad \text{and} \qquad (b, f(b))$$

does not go below the plot of f.



Convexity: Formal definition

- A function $f : \mathbb{R} \to \mathbb{R}$ is **convex** if for every choice of a, b and $t \in [0, 1]$:

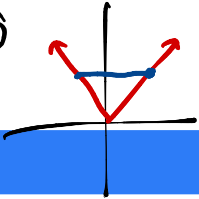
$$(1 - t)f(a) + tf(b) \geq f((1 - t)a + tb)$$

- This is a formal way of restating the condition from the previous slide.

- **We will not use this definition for anything!**
  - It's just here so that you're aware of it.

If f(x) is convex, then −f(x) is concave.

A



Discussion Question

Which of these functions is not convex?

a) $f(x) = |x|$

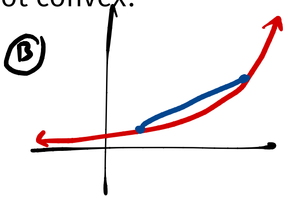
b) $f(x) = e^x$

c) $f(x) = \sqrt{x-1}$

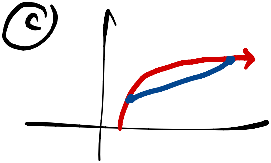
d) $f(x) = (x-3)^{24}$

To answer, go to `menti.com` **and enter the code 7910 4287.**

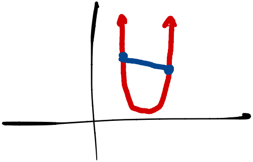
B



C



D



Why does convexity matter?

- Convex functions are (relatively) easy to minimize with gradient descent.

- **Theorem**: if $R(h)$ is convex and differentiable then gradient descent converges to a **global minimum** of R *provided* that the step size is small enough.

- **Why?**
  - If a function is convex and has a local minimum, that local minimum must be a global minimum.

  - In other words, gradient descent won't get stuck/terminate in local minimums that aren't global minimums (as happened with $R_{ucsd}(h)$ and a small σ in our demo).

Nonconvexity and gradient descent

- We say a function is nonconvex if it does not meet the criteria for convexity.
- Nonconvex functions are (relatively) hard to minimize.
- Gradient descent can still be useful, but it's not guaranteed to converge to a global minimum.
  - We saw this when trying to minimize $R_{ucsd}(h)$ with a smaller σ.

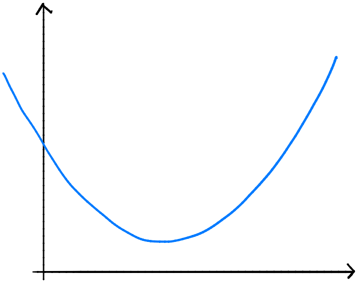
Second derivative test for convexity

- If $f(x)$ is a function of a single variable and is twice differentiable, then:

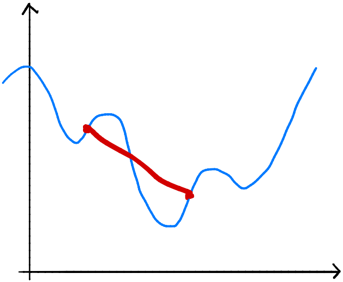
- $f(x)$ is convex if and only if $\boxed{\frac{d^2 f}{dx^2}(x) \geq 0}$ for all x.

- Example: $f(x) = x^4$ is convex.

$\frac{d}{dx} f = 4x^3$, $\frac{d^2}{dx^2} f' = 12x^2$



Convex



Non-convex

Convexity of empirical risk

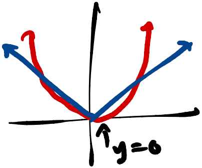
- If $L(h, y)$ is a convex function (when y is fixed) then

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

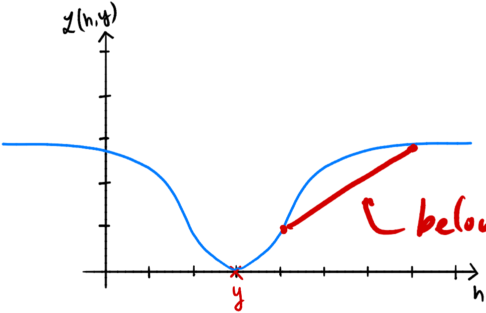
  is convex.
  - Why? Because sums of convex functions are convex.

- What does this mean?
  - If a loss function is convex (for a particular type of prediction), then the corresponding empirical risk will also be convex.

Convexity of loss functions



- Is $L_{sq}(h, y) = (y - h)^2$ convex? [✓ **Yes**] or **No**.
- Is $L_{abs}(h, y) = |y - h|$ convex? [✓ **Yes**] or **No**.
- Is $L_{ucsd}(h, y)$ convex? **Yes** or **No**.



below the function!

Convexity of R_{ucsd}

- A function can be convex in a region.

- If σ is large, $R_{ucsd}(h)$ is convex in a big region around data.
  - A large σ led to a very smooth, parabolic-looking empirical risk function with a single local minimum (which was a global minimum).

- If σ is small, $R_{ucsd}(h)$ is convex in only small regions.
  - A small σ led to a very bumpy empirical risk function with many local minimums.

Discussion Question

Recall the empirical risk for absolute loss,

$$R_{abs}(h) = \frac{1}{n}\sum_{i=1}^{n} |y_i - h|$$

Is $R_{abs}(h)$ **convex**? Is gradient descent **guaranteed** to find a global minimum, given an appropriate step size?

a) **YES** convex, **YES** guaranteed

[✓ b) **YES** convex, **NOT** guaranteed]

c) **NOT** convex, **YES** guaranteed

c) **NOT** convex, **NOT** guaranteed

To answer, go to `menti.com` **and enter the code 7910 4287.**

Summary

Summary

- Gradient descent is a general tool used to minimize differentiable functions.
  - We will usually use it to minimize empirical risk, but it can minimize other things, too.

- Gradient descent updates guesses for h^* by using the update rule

$$\boxed{h_i = h_{i-1} - \alpha \cdot \left(\frac{dR}{dh}(h_{i-1}) \right)}$$

- Convex functions are (relatively) **easy** to optimize with gradient descent.

- We like **convex loss functions**, like the squared loss and absolute loss.

What’s next?

- So far, we’ve been predicting future values (salary, for instance) without using any information about the individual.
  - GPA.
  - Years of experience.
  - Number of LinkedIn connections.
  - Major.
- How do we incorporate this information into our prediction-making process?