Lecture 6 – Simple Linear Regression

DSC 40A, Fall 2022 @ UC San Diego

Dr. Truong Son Hy, with help from **many others**

Freya Holmér
@FreyaHolmer

btw these large scary math symbols are just for-loops

Summation
(capital sigma)

$$\sum_{n=0}^{4} 3n$$

```
sum = 0;
for( n=0; n<=4; n++ )
   sum += 3*n;
```

Product
(capital pi)

$$\prod_{n=1}^{4} 2n$$

```
prod = 1;
for( n=1; n<=4; n++ )
   prod *= 2*n;
```

7:51 PM · 11 Sep 21 · Twitter Web App

Announcements

- Look at the readings linked on the course website!

- Groupwork Relsease Day: Thursday afternoon
  Groupwork Submission Day: Monday midnight
  Homework Release Day: Friday after lecture
  Homework Submission Day: Friday before lecture

- See `dsc40a.com/calendar` for the Office Hours schedule.

Agenda

- Recap of gradient descent.
- Prediction rules.
- Minimizing mean squared error, again.

Recap: gradient descent

Gradient descent

- The goal of gradient descent is to minimize a function $R(h)$.
- Gradient descent starts off with an initial guess h_0 of where the minimizing input to $R(h)$ is, and on each step tries to get closer to the minimizing input h^* by moving opposite the direction of the slope:

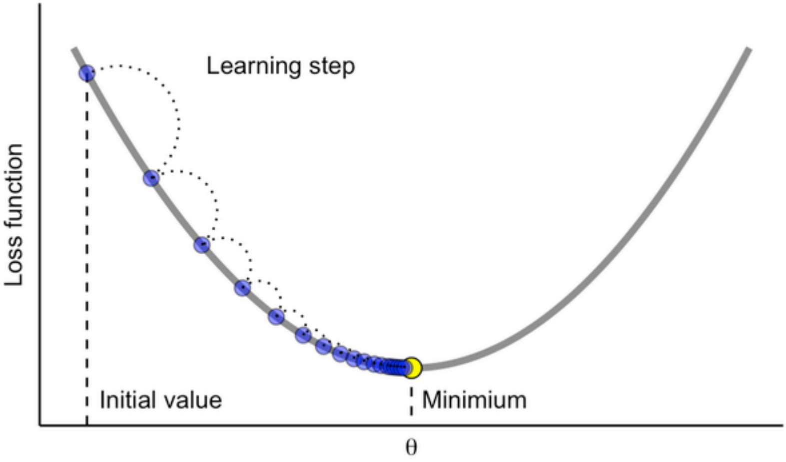
$$h_i = h_{i-1} - \alpha \cdot \frac{dR}{dh}(h_{i-1})$$

  - α is known as the learning rate, or step size. It controls how much we update our guesses by on each iteration.
- Gradient descent terminates once the guesses h_i and h_{i-1} stop changing much.

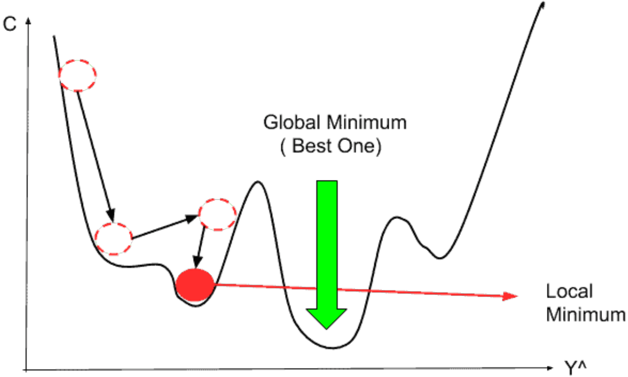
You will not be responsible for implementing gradient descent in this class, but here's an implementation in Python if you're curious:

```
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
```

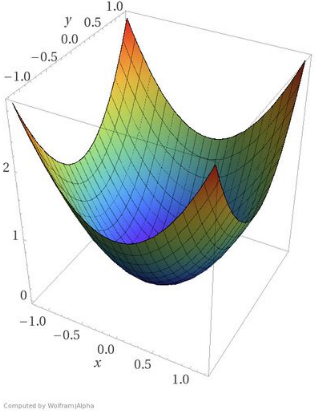
Gradient descent (convex loss)

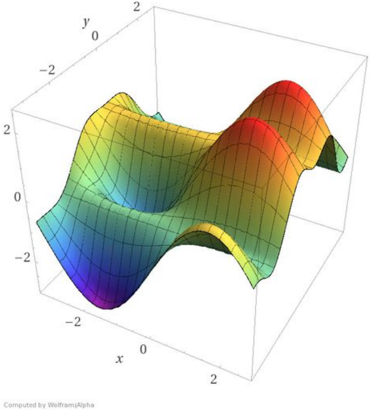


Local minimum (Non-convex loss)



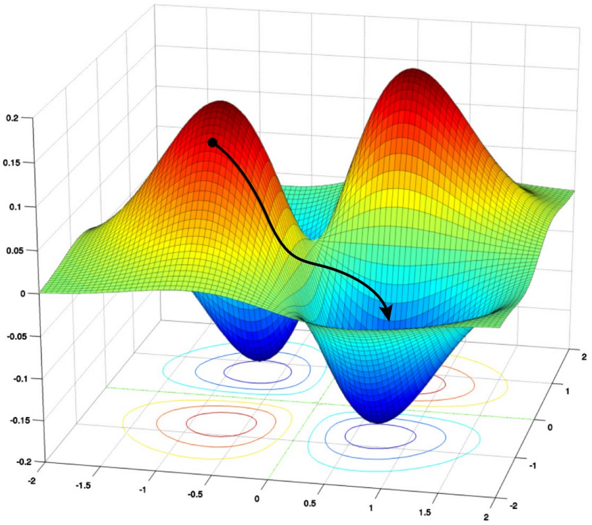
Convex vs. Non-convex (higher dimensions)



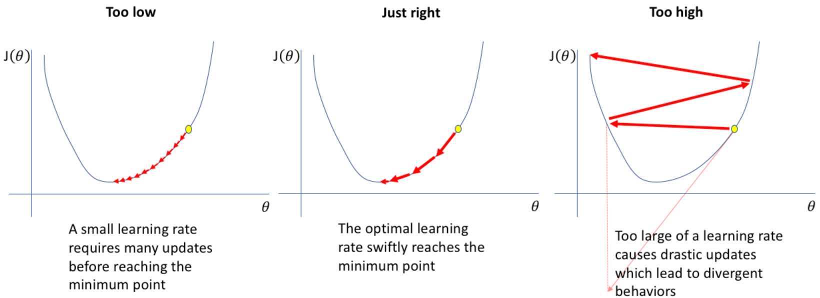


Convex & Non-convex

Gradient descent in higher dimensions



Problem with learning rates



A small learning rate requires many updates before reaching the minimum point

The optimal learning rate swiftly reaches the minimum point

Too large of a learning rate causes drastic updates which lead to divergent behaviors

Why does convexity matter?

- **Gradient descent**:

$$h_i = h_{i-1} - \alpha_i \cdot \frac{dR}{dh}(h_{i-1})$$

  where α_i is the learning rate at step i-th.

- **Theorem** (informal): 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 (i.e. $\lim_{i \to +\infty} \alpha_i = 0$).

- **Why?**
  Convex functions are (relatively) easy to minimize with gradient descent. If a function is convex and has a local minimum, that local minimum must be a global minimum.

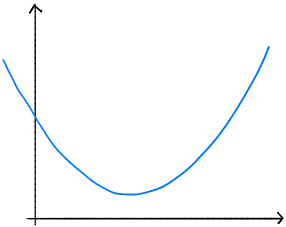
Nonconvexity and gradient descent

- 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.

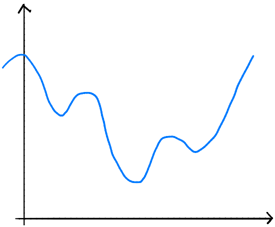
Second derivative test for convexity

- If $f(x)$ is a function of a single variable and is twice differentiable, then: $f(x)$ is convex if and only if $\frac{d^2 f}{dx^2}(x) \geq 0$ for all x.

- A twice-differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if the **Hessian** $\nabla^2 f(x) \in \mathbb{R}^{n \times n}$ **is positive semi-definite** at every $x \in \mathbb{R}^n$.



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 $L_{\text{sq}}(h, y) = (y - h)^2$ convex?

Convexity of loss functions

- Is $L_{\text{sq}}(h, y) = (y - h)^2$ convex? **Yes**.
- Is $L_{\text{abs}}(h, y) = |y - h|$ convex?

Convexity of loss functions

- Is $L_{\text{sq}}(h, y) = (y - h)^2$ convex? **Yes**.
- Is $L_{\text{abs}}(h, y) = |y - h|$ convex? **Yes**.
- Is $L_{ucsd}(h, y)$ convex?

Convexity of loss functions

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

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

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

Answer: A. **Mostly!** We have to care about where we cannot compute the derivative.

When does gradient descent work?

- A function f is convex if, for any two inputs a and b, the line segment connecting the two points $(a, f(a))$ and $(b, f(b))$ does not go below the function f.
  - $R_{abs}(h) = \frac{1}{n} \sum_{i=1}^{n} |y_i - h|$: convex.
  - $R_{sq}(h) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h)^2$: convex.
  - $R_{ucsd}(h) = \frac{1}{n} \sum_{i=1}^{n} \left[1 - e^{-(y_i - h)^2 / \sigma^2}\right]$: not convex.
- **Theorem:** If $R(h)$ is convex and differentiable then gradient descent converges to a **global minimum** of R given an appropriate step size.

Prediction rules

How do we predict someone's salary?

After collecting salary data, we...

1. Choose a loss function.

2. Find the best prediction by minimizing empirical risk.

- So far, we've been predicting future salaries without using any information about the individual (e.g. GPA, years of experience, number of LinkedIn connections).

- **New focus:** How do we incorporate this information into our prediction-making process?

Features

A **feature** is an attribute – a piece of information.

- **Numerical**: age, height, years of experience
- **Categorical**: college, city, education level
- **Boolean**: knows Python?, had internship?

Think of features as columns in a DataFrame (i.e. table).

	YearsExperience	Age	FormalEducation	Salary
0	6.37	28.39	Master's degree (MA, MS, M.Eng., MBA, etc.)	120000.0
1	0.35	25.78	Some college/university study without earning ...	120000.0
2	4.05	31.04	Bachelor's degree (BA, BS, B.Eng., etc.)	70000.0
3	18.48	38.78	Bachelor's degree (BA, BS, B.Eng., etc.)	185000.0
4	4.95	33.45	Master's degree (MA, MS, M.Eng., MBA, etc.)	125000.0

Variables

- The features, x, that we base our predictions on are called **predictor variables**.
- The quantity, y, that we're trying to predict based on these features is called the **response variable**.
- We'll start by predicting salary based on years of experience.

Prediction rules

- We believe that salary is a function of experience.
- In other words, we think that there is a function H such that:

$$\text{salary} \approx H(\text{years of experience})$$

- H is called a **hypothesis function** or **prediction rule**.
- **Our goal**: find a good prediction rule, H.

Possible prediction rules

$$H_1(\text{years of experience}) = \$50{,}000 + \$2{,}000 \times (\text{years of experience})$$

$$H_2(\text{years of experience}) = \$60{,}000 \times 1.05^{(\text{years of experience})}$$

$$H_3(\text{years of experience}) = \$100{,}000 - \$5{,}000 \times (\text{years of experience})$$

- These are all valid prediction rules.
- Some are better than others.

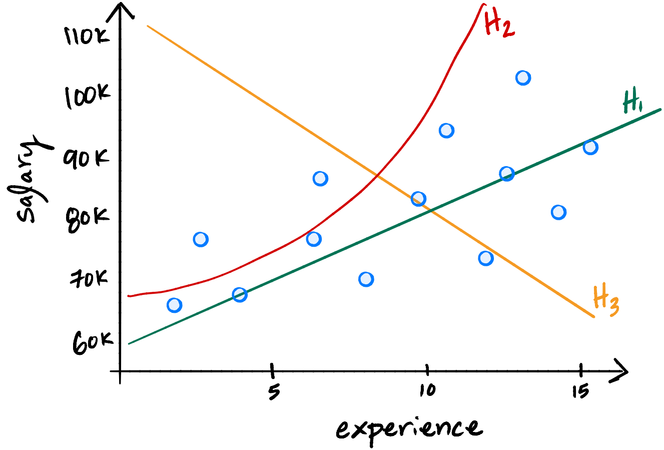
Comparing predictions

- How do we know which prediction rule is best: H_1, H_2, H_3?
- We gather data from n people. Let x_i be experience, y_i be salary:

$$\begin{array}{c} (\text{Experience}_1, \text{Salary}_1) \\ (\text{Experience}_2, \text{Salary}_2) \\ \dots \\ (\text{Experience}_n, \text{Salary}_n) \end{array} \quad \to \quad \begin{array}{c} (x_1, y_1) \\ (x_2, y_2) \\ \dots \\ (x_n, y_n) \end{array}$$

- See which rule works better on data.

Example

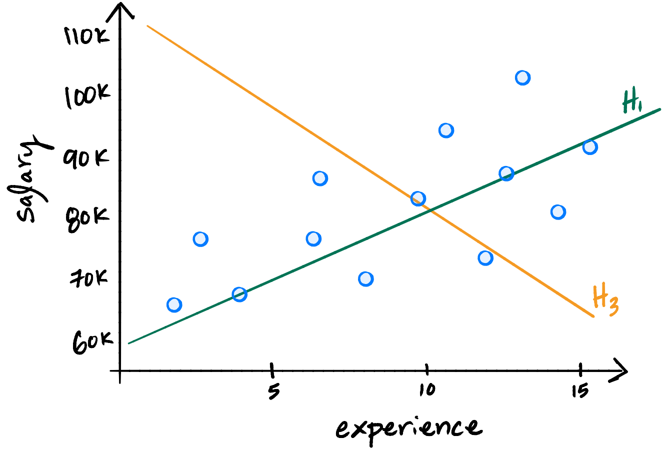


Quantifying the quality of a prediction rule H

- Our prediction for person i's salary is $H(x_i)$.
- As before, we'll use a **loss function** to quantify the quality of our predictions.
  - Absolute loss: $|y_i - H(x_i)|$.
  - Squared loss: $(y_i - H(x_i))^2$.
- We'll use squared loss, since it's differentiable.
- Using squared loss, the **empirical risk** (mean squared error) of the prediction rule H is:

$$R_{sq}(H) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - H(x_i)\right)^2$$

Mean squared error

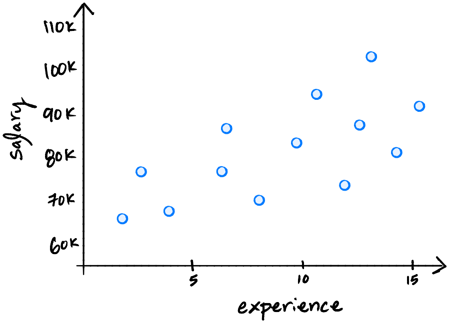


Finding the best prediction rule

- **Goal:** out of all functions $\mathbb{R} \to \mathbb{R}$, find the function H^* with the smallest mean squared error.

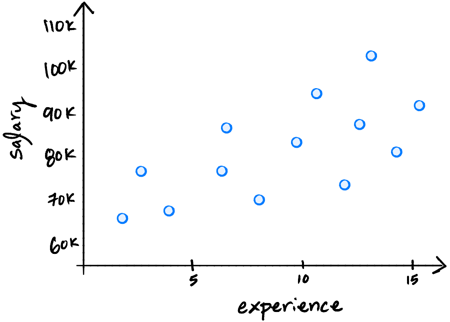
- That is, H^* should be the function that minimizes

$$R_{sq}(H) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - H(x_i)\right)^2$$



Discussion Question

Given the data below, is there a prediction rule *H* which has **zero** mean squared error? Yes or No?



Discussion Question

Given the data below, is there a prediction rule H which has **zero** mean squared error? Yes or No?

Answer: Yes! That is bad! Why?

Next time: We will learn more about the choice of H.