Bisection search, Continuous optimization

6.100 LECTURE 7

SPRING 2026

Announcements

- Snow day today
  - office hours are remote, enter a Zoom link on the queue
- Today's lecture format
  - stay muted, camera off, ask questions in chat
  - be patient if technical issues
- **Exam 1** this Wednesday 2/25 in Walker 50-340
  - no instructor office hours this week
- Pset 3 released Wednesday after the exam
- Pset 2 due tonight
- Pset 2 checkoffs start tomorrow

Float representation and precision

Binary representation

- Numbers are stored in memory using bits, hence base 2 representation
  - 4_{10} → 100_2
  - 10_{10} → 1010_2
  - $(1/8)_{10}$ → 0.001_2
  - $(9/5)_{10}$ → $1.1100\ 1100\ 1100\ ..._2$
- If a real number cannot be represented as a fraction with an exact power of 2 in the denominator, its **`float`** representation must be an approximation
- Hence, repeated operations on **`float`**s may result in rounding errors
- Details of float representation
  - https://en.wikipedia.org/wiki/IEEE_754

Guidelines for working with `floats`

- Don't rely on float values to exactly represent real numbers
- Don't stress about representation-level inaccuracy
  - precision is usually good enough (around 16 decimal digits)
  - just be aware when dealing with wide range of magnitudes
- **Avoid == comparisons on floats,** alternatives:
  - `result < bound`
  - `abs(result - expected) <= tolerance`
  - `math.isclose()`
- Don’t search for exact answer, define tolerance instead

Finding an approximate answer

Exhaustive enumeration

- Consider problem of finding the square root of a number **`x`**
  - (without using **`**0.5`** or **`math.sqrt()`**)
- Can’t guarantee (and most likely impossible) to find exact float representation
- But can define **`epsilon`** tolerance (e.g., 0.001) in the answer

Exhaustive enumeration

- Consider problem of finding the square root of a number **`x`**
  - (without using **`**0.5`** or **`math.sqrt()`**)
- Can’t guarantee (and most likely impossible) to find exact float representation
- But can define **`epsilon`** tolerance (e.g., 0.001) in the answer
- **First strategy: exhaustive enumeration**
  - step through the values 0, 0.001, 0.002, 0.003, ..., **`x`**
    - use **`numpy.arange()`**
  - record how close the square of each one is to **`x`**
  - return the value with the smallest error
- **Issues**
  - waste time checking candidates far past the answer
  - takes 10 times longer for every decimal point of precision
  - do we ned to check candidates close to zero?

Bisection search

- **Insight:** the square root of a (non-negative) **x** increases as **x** increases, i.e., the relationship is ***monotonic***
  - if `guess**2 > x`, then `guess` is greater than the true square root
  - if `guess**2 < x`, then `guess` is less than the true square root
- **Idea:** check the middle and discard half of the solution space
  - start with bounds within which the solution is guaranteed
  - guess the midpoint of the bounds
  - if infer guess is too large, cut out the upper half
  - otherwise cut out the lower half
  - repeat and stop when remaining solution space is tighter than desired precision

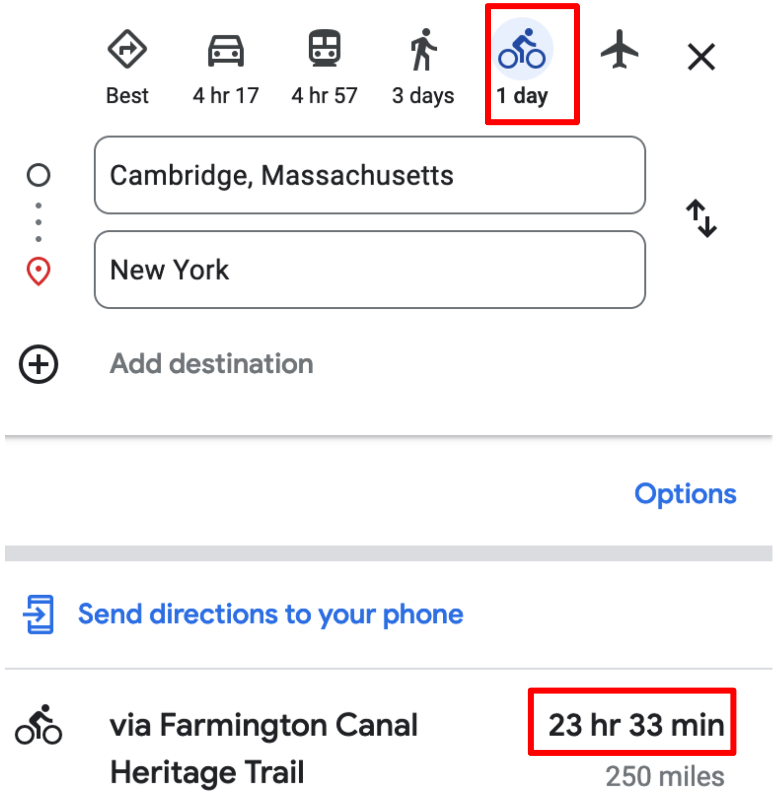
Bisection search properties

- For each decimal point of precision, need only about ***3 additional iterations***
  - not 3 times longer!
- **Alternate exit condition:** look for error **`delta`** with respect to ***input query value*** instead of output answer
  - i.e., distinguish between square roots of 100.0 and 100.1
  - larger queries demand greater precision in output when
- Works for computing the **inverse of any monotonic relationship**
  - square root, cube root, logarithm, etc.
- Also works for ***ordered*** discrete spaces
  - e.g., a sorted sequence of words, a sorted list of outcomes
  - but need to careful about indexing

Optimization problems

What is optimization?

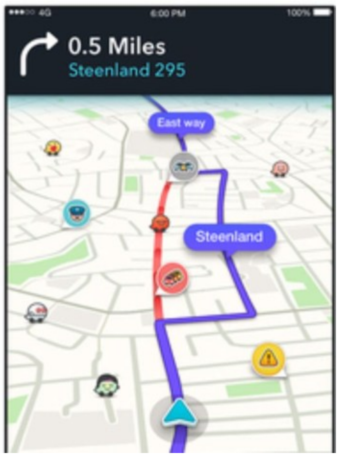
- An ***objective function*** that is to be maximized or minimized
  - e.g., minimize money spent traveling from Cambridge to NYC
- Optionally (but often) a ***set of constraints***
  - e.g., expected transit time < 5 hours

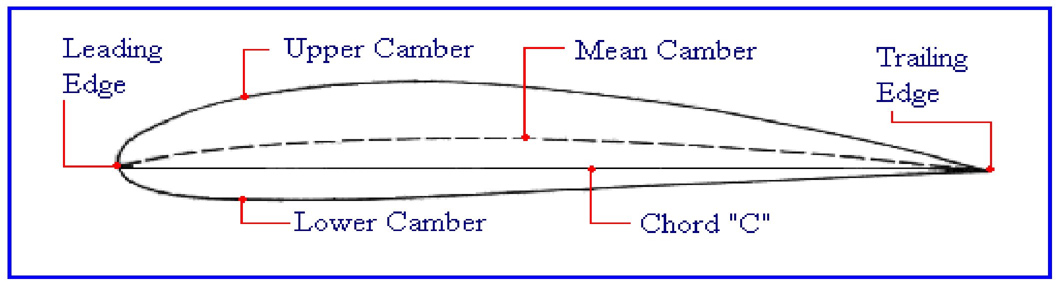


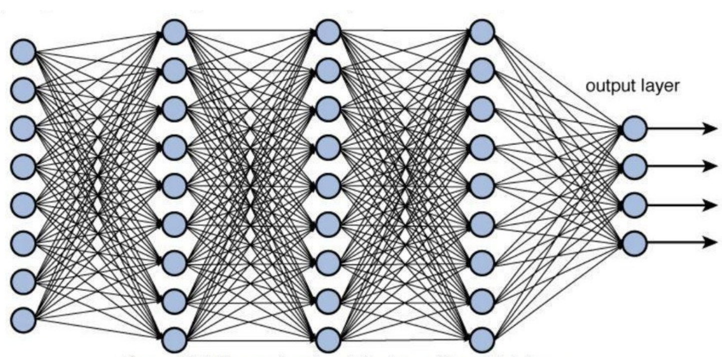
If you can average 10.6 mph for 23.5 hours, congratulations!

Optimization problems

- Anytime you are trying to maximize or minimize something, you are solving an optimization problem







prediction

cat

bird

Minimize difference

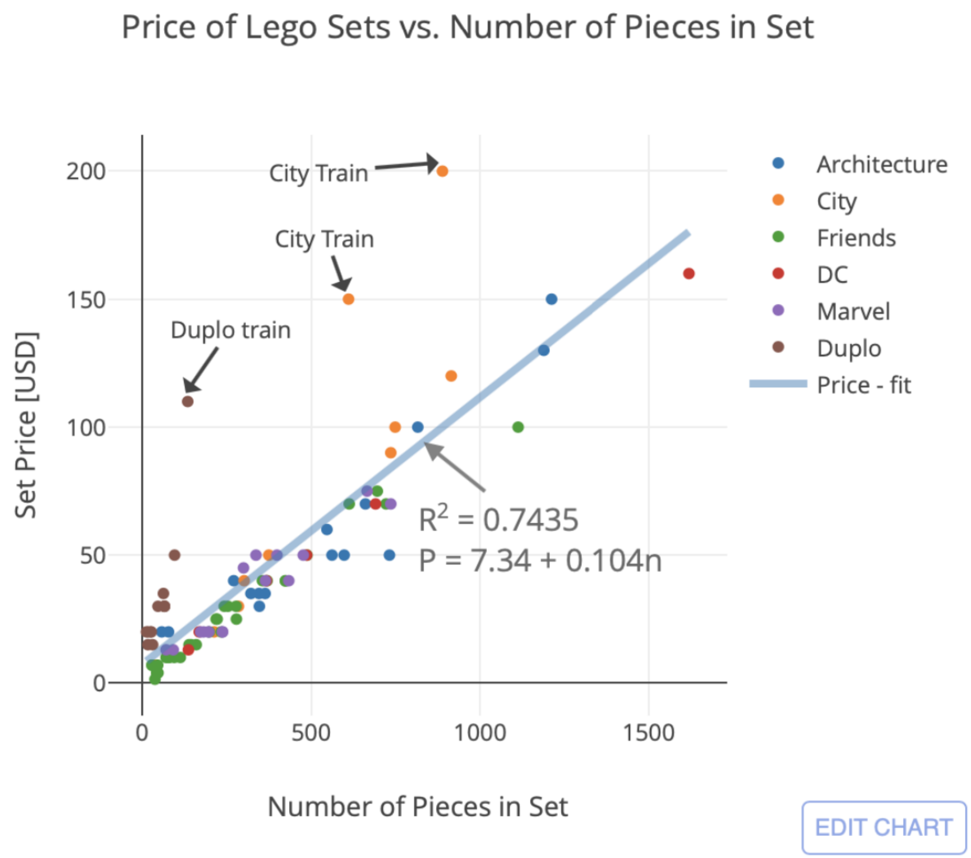
ground truth label

Solving optimization problems

- Objective and constraints are defined over variables
  - continuous variables – e.g., driving speed
  - discrete variables – e.g., whether to take highway or local roads ("two roads diverged in the yellow wood...")
- Methods differ depending on whether variables or continuous or discrete or both (hybrid/mixed)
- Often easier to solve for continuous variables
  - if objectives and constraints are differentiable, can take advantage of that

Curve-fitting

- Distributions alone lack the power to explain relationships between random variables



https://www.wired.com/2014/08/lego-cost

Linear regression

- Given a set of (x, y) data points
  - $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Want to model a predictive relationship between them
  - i.e., predict $\boldsymbol{y}$ value associated with any $\boldsymbol{x}$
- Simplest relationship is $\boldsymbol{y} = \boldsymbol{a} \cdot \boldsymbol{x} + \boldsymbol{b}$
  - specifies a linear model
- The task is to find parameters $\boldsymbol{a}$ and $\boldsymbol{b}$ that "best fit" the data
  - $\boldsymbol{a}$ indicates upward or downward trend and how much
- Can we frame this as an optimization problem?

Defining an objective

- Consider a given model $\boldsymbol{y = a \cdot x + b}$
- For each data point $(\boldsymbol{x_i}, \boldsymbol{y_i})$, the predicted $\boldsymbol{y}$ value is
$$\boldsymbol{\hat{y}_i = a \cdot x_i + b}$$
- For distributions, we've discussed **variance**
  - average of squares of differences against mean
  - larger variance means less certainty about a sample's value
- To evaluate how well we're predicting $\boldsymbol{y}$ overall, we should compare each $\boldsymbol{\hat{y}_i}$ against its corresponding $\boldsymbol{y_i}$, not just the overall mean $\boldsymbol{\mu_y}$ of all $\boldsymbol{y_i}$'s

Linear regression as optimization

- Consider the **sum of all squared errors**

$$\sum_{i=1}^{n}(\hat{y}_i - \boldsymbol{y_i})^2 = \sum_{i=1}^{n}(\boldsymbol{a} \cdot \boldsymbol{x_i} + \boldsymbol{b} - \boldsymbol{y_i})^2$$

- The average of this is the **mean squared error (MSE)**
- This is a quadratic expression in terms of $\boldsymbol{a}$ and $\boldsymbol{b}$
  - the $\boldsymbol{x_i}$ and $\boldsymbol{y_i}$ values are known, so they are constants
  - the number of data points $\boldsymbol{n}$ is also a constant
  - $\boldsymbol{a}$ and $\boldsymbol{b}$ are **continuous variables** in our objective function
- There exist efficient (even closed-form) solutions for the optimal $\boldsymbol{a}$ and $\boldsymbol{b}$ that minimize the MSE
  - available in **numpy,** will show next lecture
  - Pset 3 begins by asking you to implement exhaustive enumeration and bisection approaches

Next time

- **Recitation, Fri 2/27**
  - recap random walks, CLT, bisection search
- **Lecture 9, Mon 3/2**
  - generalize regression to polynomial models
  - determining appropriate model complexity
- **Lecture 10, Wed 3/4**
  - revisit sample mean and CLT
  - making valid statistical claims

- **Good luck on Exam 1!**