Computational Structures in Data Science

Lecture #1: Welcome to CS88!

UC Berkeley EECS
Adj. Ass. Prof. Dr. Gerald Friedland

January 28, 2019

http://net.eecs.berkeley.edu/~cs88

CS88 - me

Projects you might want to check out:

- http://mmcommons.org
  - Work with 100M images, 1M videos in your own Amazon instance.

- http://www.teachingprivacy.org
  - Creating teaching materials informing about data over sharing.

CS88 Team

Teaching Assistants

CS88 Team - me

- Dr. Gerald Friedland (fractor@berkeley.edu)
  - 424 Saturday Daj Hall (CITRIS)
  - http://www.gerald-friedland.org
  - Office hours: Mo 1:30-2:30 @ 424 SDH
  - Before/after class

- Adjunct Assistant Professor, EECS UC Berkeley
  - Principal Data Scientist, Lawrence Livermore National Laboratories

Goals today

- Introduce you to
  - the field
  - the course
  - the team
- Answer your questions
- Big Ideas:
  - Abstraction
  - Data Type
Data Science

Nearly every field of discovery is transitioning from “data poor" to "data rich."  

A connected world

Computational Thinking + Inferential Thinking in the context of working with real world data

Data 8 – Foundations of Data Science

- Deeper understanding of the computing concepts introduced in c8
  - Hands-on experience => Foundational Concept
  - How would you create what you use in c8?
- Extend your understanding of the structure of computation
  - What is involved in interpreting the code you write?
  - Deeper CS Concepts: Recursion, Objects, Classes, Higher-order Functions, Declarative programming, …
  - Managing complexity in creating larger software systems through composition
- Create complete (and fun) applications
- In a data-centric approach

Pathways

- c8
- cs88
- c8 cs88 cs47a cs61b
- *** CS major
- c8
- cs61a
- cs61a
How does CS88 relate to CS61A?

<table>
<thead>
<tr>
<th>Units</th>
<th>CS61A</th>
<th>CS/INFO/STAT c8</th>
<th>CS88</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro Programming &amp; Tools</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS Concepts and Techniques</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistics Concepts in a Computational Approach</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working w/ Data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thinking w/ Data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Course Structure

- 1 Lecture + 1 Lab/Discussion on Wednesday (!!!)
- Lecture introduces concepts (quickly!), answers why questions.
- Lab provides concrete detail hands-on
- Homework (10) cements your understanding
  - Out Monday, Due Sunday
- Projects (3) put your understanding to work in building complete applications
  - Maps
  - Hangman
  - Open Projects!
- Readings: [http://composingprograms.com](http://composingprograms.com)
  - Same as cs61a

Course Culture

- Learning
- Community
- Respect
- Collaboration
- Peer Instruction

Where will we work?

- Datahub.berkeley.edu
- Your laptop
- Inst.eecs.Berkeley.edu

Piazza for {ask,answer}ing questions

Pro-student Grading Policies

- EPA
  - Rewards good behavior
  - Effort
    - E.g., Office hours, doing every single lab, hw, reading Piazza pages
  - Participation
    - E.g., Raising hand in lec or discussion, asking questions on Piazza
  - Altruism
    - E.g., helping other students in lab, answering questions on Piazza
- You have 2 “Slip Days”
  - You use them to extend due date, 1 slip day for 1 day extension
  - You can use them one at a time or all at once or in any combination
  - They follow you around when you pair up (you are counted individually)
    - E.g., A has 2, B has 0. Project is late by 1 day. A uses 1, B is 1 day late
**Abstraction**

- **Detail removal**
  “The act of leaving out of consideration one or more properties of a complex object so as to attend to others.”

- **Generalization**
  “The process of formulating general concepts by abstracting common properties of instances”

- **Technical terms:** Compression, Quantization, Clustering, Unsupervised Learning

---

**Experiment**

---

**Where are you from?**

Possible Answers:

- China
- California
- The Bay Area
- San Mateo
- 1947 Center Street, Berkeley, CA
- 37.8693° N, 122.2696° W

All correct but different levels of abstraction!

---

**Abstraction gone wrong!**

---

**Detail Removal (in Data Science)**

- You’ll want to look at only the interesting data, leave out the details, zoom in/ out...
- Abstraction is the idea that you focus on the essence, the cleanest way to map the messy real world to one you can build
- Experts are often brought in to know what to remove and what to keep!

---

**The Power of Abstraction, Everywhere!**

- **Examples:**
  - Functions (e.g., sin x)
  - Hiring contractors
  - Application Programming Interfaces (APIs)
  - Technology (e.g., cars)

- Amazing things are built when these layer
  And the abstraction layers are getting deeper by the day!

---

The London Underground 1928 Map & the 1933 map by Harry Beck.

---

The Power of Abstraction, Everywhere!

- **Example:**
  - Functions (e.g., sin x)
  - Hiring contractors
  - Application Programming Interfaces (APIs)
  - Technology (e.g., cars)

- Amazing things are built when these layer
  - And the abstraction layers are getting deeper by the day!

---

We only need to worry about the interface, specification, or contract
NOT how (or by whom) it's built

**Above the abstraction line**

- Abstraction Barrier (interface)
  (the interface, or specification, or contract)

**Below the abstraction line**

This is where / how / when / by whom it is actually built, which is done according to the interface, specification, or contract.
Abstraction: Pitfalls

• Abstraction is not universal without loss of information (mathematically provable). This means, in the end, the complexity can only be “moved around”

• Abstraction makes us forget how things actually work and can therefore hide bias. Example: AI and hiring decisions.

• Abstraction makes things special and that creates dependencies. Dependencies grow longer and longer over time and can become unmanageable.

Abstraction in CS: Data Type

• What’s this?

![](image)

- Real (or ideal) world
- Computer representation

Data Types and Operations

• Set of elements
  - with some internal representation
  - e.g. Integers, Floats, Booleans, Strings, ...

• Set of operations on elements of the type
  - e.g. *, /, +, %, //, **
  - ==, <, >, <=, >=

• Properties
  - Commutative, Associative, ...

• Expressions are valid well-defined sets of operations on elements that produce a value of a type

Lab and HW this week

• Lab will get you to where you have a program development environment
  - Even on your computer

• HW will give practice and explain subtleties of types, operators, and expressions
  - In a program development environment

• What’s the difference between ‘==’ and ‘=’?

Thoughts for the Wandering Mind

A binary digit (bit) is a symbol from {0,1}.

• How many strings can you represent with N bits?

• Could you build a program that compresses all strings of N bits to strings of M bits (with M < N) such that you can go back to all original strings of length N? How or Why?