CLASS DESCRIPTIONS— SATURDAY 7/22, MATHCAMP 2017 REUNION!

Contents

Rooms: T = Thompson, J = Jones, M = McIntyre
Blurbs for regular Mathcamp Week 3 classes are in a separate packet.

9:10 Classes

Quantum Parallelism: Computing a Black-Box Function with the Minimal Number of Queries. (*Andrew Guo*, 🌶🌶🌶, T310)
In 1985, David Deutsch came up with the first quantum algorithm to solve the following problem: given access to a black-box that implements an unknown binary function, determine whether the function is constant or not. While the problem itself is trivial, the implications of the algorithm were momentous: Deutsch proved that his algorithm required strictly fewer "queries" to the black-box than *any* classical deterministic algorithm! His premier result in the field of quantum algorithms hinted at the possibility that, someday, quantum computers could solve certain problems more efficiently than any classical computer. Indeed, he opened the door for Peter Shor's seminal discovery of a polynomial-time algorithm for integer factorization in 1994. If implemented on a large-scale quantum computer, Shor's algorithm would pose an existential threat to the security of RSA encryption! In this class, I will introduce you to Deutsch's algorithm, the de facto "Hello, World!" of quantum computation. We will not assume any prior exposure to quantum physics or the existence of numbers greater than two.

Prerequisites: Linear Algebra

Who is Andrew? Andrew Guo was a camper at MC'08, '10-'12 and a JC in '14. He is now a graduate student in Physics at the University of Maryland, College Park. He believes in the Church-Turing Thesis, but not in the stronger, "extended" version.

Using Math to Protect Democracy. (*Mira Bernstein*, 🌶, M103)

Note: *This is the same talk that I gave as a colloquium on Day 2, so MC17 campers shouldn't come to this class! But anyone who wants to know more about this subject shoulc come to*

Saturday's Voting Theory Superclass (at 11:10 and 1:10), which will be a direct continuation of this talk.

Every 10 years, the US has a census to determine the number of representatives each state should get in Congress. Then the legislature in each state comes up with a *districting plan*: a way of splitting the state into the correct number of regions (districts) of equal population, each of which will elect one representative to Congress.

The problem is, the specific choice of districting plan can have a huge effect on the results. For instance, suppose a state of 400,000 people is allotted four representatives. There are two political parties, A and B, and Party B happens to be in power at the time of the census. It does some polling and adopts the following districting plan:

District	A supporters	B supporters
1	85,000	15,000
2	45,000	55,000
3	45,000	55,000
4	45,000	55,000

Now B can count on winning 3 out of 4 districts in the next election, even though 55% of the the voters in the state support A!

This kind of thing happens all the time; both Republicans and Democrats do it. It seems so obviously wrong that you wonder how it can be legal. But it turns out that determining whether a districting plan is "fair" or "neutral" is much more complicated than you might think. Right now, many people, including lawyers, political scientists, computer scientists, and mathematicians are working hard to figure out a solution to the gerrymandering problem before the next census in 2020. You can help! Come to this talk to learn how.

Who is Mira? Mira has been at Mathcamp every year since 1997 (over two years total). She has never been a camper or (officially) a JC, but she's been a mentor, AC, faculty, Top Banana, and now Top Banana Emerita. While not at Mathcamp, she splits her time between cool math education start-ups (like Proof School and BEAM) and data-science-for-good projects (like the Metric Geometry and Gerrymandering Group at Tufts, of which she is a founding member). She once got charged by a hawk while playing "Nonabelian" on her ocarina.

10:10 Classes

Why Computers Don't Read Good but Do Other Stuff Good. (*Greg Burnham*, 🌶, T310)
Human language is tantalizingly close to a formal system. We *feel* like there is a clear relationship between the words we express and the information these words convey. At worst, we just need to be a little more verbose and explicit. But if this intuition is true, then we should be able to write computer programs to perform linguistic tasks – like reading a document and answering questions about it. But we've been trying to write such programs for 50 years, and the results are mixed at best.

This class will be a quick survey of some interesting topics in the (very broad) field of computational natural language understanding. We'll try to motivate why it's so difficult to write computer programs capable of performing linguistic tasks and then describe what tasks computers *can* currently perform, focusing on how recent algorithmic and technological progress has allowed for improved performance. We will conclude by noting that the big problems remain unsolved and speculating on what might be necessary for the next steps forward.

As a teaser, here is my favorite short example motivating why computational language understanding is hard. Consider the following two sentences, which differ only in the last word:
"The cat caught the mouse because it was clever."
"The cat caught the mouse because it was careless."
Now, what does the pronoun "it" refer to in each sentence? Humans share a clear intuition about the right answer to this question. And yet, consider what it would take to write a computer program with this same capability. That's the problem in a nutshell.

Who is Greg? Greg was a camper in '04-'06 and a JC in '07, '08, '10. He currently does AI research in language understanding at a small lab called Elemental Cognition. On Monday he will be on the NPR quiz show Ask Me Another, where his bio fun-fact will probably be about Mathcamp.

Social Justice and Math Education. (*Dan Zaharopol*, 🌶, M103)
It is an uncomfortable truth that high-level math programs (including Mathcamp) simply don't have many non-Asian students of color, nor do we have a representative sample of low-income students. This is despite an exceptionally generous scholarship program and an admissions process based on demonstrated work output. So, how can we open up pathways to success in science, mathematics, engineering, and programming to historically underserved communities so that they are prepared for college majors and places such as Mathcamp?

Six years ago, I started Bridge to Enter Advanced Mathematics (BEAM), a program that has been working to close exactly these gaps. Several of our students have continued to high-level opportunities and we've learned a lot about helping students access advanced work. I'll share lessons learned and some of the crazy stories from the years of the program.

Who is Dan? Dan has held just about every role at Mathcamp: camper in 1999–2000, JC in 2001–2004, mentor in 2005 – 2008, faculty in 2009 – 2010, and visitor in 2011 – 2016. He stopped coming to Mathcamp full time when he began running "Dancamp": Bridge to Enter Advanced Mathematics (BEAM), a summer program for underserved students with talent in math. He hates it when people call him Mr. Zaharopol or steal his shampoo; that doesn't mean you shouldn't try.

11:10 Classes

Linear Logic. (Anti Shulman, 🌶🌶🌶, T310)
When we learn mathematics, the underlying logic is usually taken as a given. But over the years, many mathematicians, philosophers, and scientists have raised various problems with this logic, such as:

(1) Its conclusions need not be *relevant*: statements like “if my hair is purple then the sun rises in the east” are considered true.
(2) It is *explosive*: statements like “if $1 = 2$ then I am the Pope” are also considered true.
(3) It is not *resource-sensitive*: from “if I have \$1 then I can buy chips” and “if I have \$1 then I can buy a drink” it lets us conclude (incorrectly) that “if I have \$1 then I can buy chips and a drink”.
(4) It is not *constructive*: a proof that something exists doesn’t give us a way to find that thing.
(5) It is not *effective*: even if we have an algorithm to find something in principle, the universe might die before we actually find it.
(6) It does not apply in the *quantum* world of subatomic particles.
(7) When applied naïvely, it leads to *contradictions* such as Russell’s paradox.

Linear logic is a different kind of logic that solves *all* of these problems!

Prerequisites: Some exposure to formal mathematical logic.

Who is Anti? Anti (short for “Antimike”, Mike in the real world) was a camper at MC 96 - 98, a JC at MC 01 and 02, a Mentor at MC 03-06, an AC/AO in MC 07 - 08, and a part-time visiting faculty in a few other years. Over one year of Anti’s has been spent at Mathcamp, and he met his future spouse Megan when they were both JCs at Mathcamp 2001. When not visiting Mathcamp, he works on revolutionizing the foundations of mathematics and physics using formalized constructive homotopy type theory. Set theory, pencil and paper, and classical logic are so 20th century; nowadays the basic objects of mathematics are ∞-categories, we can formalize it using a computer, and we can design new kinds of logic whenever we need them.

Bill’s Favorite 5-Minute Math Tricks. (*Bill Kuszmaul*, 🌶🌶🌶, T399)
I collect five-minute math snippets to share with my friends. For this class, I’ll present a few of my favorites, ranging from probabilistic paradoxes to the Soviet Russian version of Cantor’s diagonalization argument.

Prerequisites: Some familiarity with probability.

Who is Bill? Bill was a camper in 2012 and 2013, and has also written a whole bunch of problems for the Mathcamp Qualifying Quiz. Currently he is an undergraduate at Stanford studying math and CS. He can count in binary on his fingers.

Proving (a tiny bit of) the Internet Secure. (*Jason Gross*, 🌶🌶, M103, 11:10 - 11:30)
Cake Cutting: Theory and Practice. (*Misha Lavrov*, 🌶, M103, 11:35 - 12:00)

Note: *These two short classes are running in the same space and time slot.*

Jason’s blurb: Designing Internet security involves three steps:
(1) Find a really hard math problem.
(2) Make it so that in order to steal your data, attackers have to solve that problem.
(3) Compile the function for securing data from math to code.

This third step is surprisingly hard, and the smallest mistake reveals all your data to the world. Come learn why mistakes are so easy, and what it takes to build a system for compiling math to code that makes most such mistakes impossible!

Misha's blurb: In this class, we will discuss how n people can fairly split a cake, what "fairly" even means, and what happens if the cake eaters get picky about pesky details like "having a continuous piece of cake" and "getting a piece of cake they can reach". There will be cake.

Who is Jason? Jason loved being a camper MC06–MC09 (inclusive). He's currently having an awesome time working on proving the Internet secure, and getting computers to automatically turn math into code (with correctness proofs!), as part of his PhD research at MIT. He's flying to the reunion straight from Israel, where he (intentionally) fell out of a plane, solved a Rubik's cube on the way down, and then went scuba diving, and snorkeling with dolphins.

Who is Misha? Misha was a camper at MC06 and MC07, then returned as a mentor for 2014, 2015, 2016, and 2017. When not at camp, he is a postdoc at the University of Illinois Urbana-Champaign. He specializes in getting campers to hit each other with pillows.

1:10 Classes

A Countable Hat Game That You Can and Can't Win. (*Jalex Stark*, 🌶🌶🌶, T310)
A countable infinity of players play a game in which they have to guess the colors of hats that they can't see. (We'll define the game precisely in the class.) Do they have a strategy which always lets them win? Yes and no.

We'll first prove a result by Gabay and O'Connor: Assuming the axiom of Choice, they have a strategy that always win. Then we'll prove a folklore[1] result: Assuming that all sets of reals are measurable, there is no strategy that lets them win with probability 1.

Prerequisites: Definition of a measure space

Who is Jalex? Jalex was a camper in 2012 and 2013, a JC in 2014 and 2015, and is also a JC in 2017. They're currently researching quantum computer science at Caltech. At the 2015 all-nighter, campers put 142 pencils in Jalex's hair. It was fun!

Building Machines that Learn and Think Like People. (*Josh Tenenbaum*, 🌶🌶, M103)
I will talk about some of the latest work that our group is doing at MIT on reverse-engineering human intelligence in computational terms, and building artificial intelligence systems that are smarter in more human-like ways. I will discuss the differences between this human-inspired approach to AI and much of the work at the center of the current "deep learning" movement. We use methods based on probabilistic programs, Bayesian inference and program induction, and simulation tools from video game engines, to capture how people learn new concepts from just one or a few examples (as opposed to hundreds or thousands of examples, as in conventional machine learning), and also to capture how people can reason flexibly about the physical world and the mental states (beliefs and desires, or "theory of mind").

Who is Josh? Josh has been a visitor at Mathcamp almost every year since 1998. As a professor of Brain and Cognitive Science at MIT, he builds probabilistic models of human cognition and tries to reverse engineer the mind. Lots of psychologists try to show that people are, in general, stupider than they seem; Josh tries to show that they are actually smarter than they seem!

[1]A result similar to this appears in Hardin and Taylor's book "The Mathematics of Coordinated Inference".