1 My Perspective

I'd describe my interests as power-law-distributed in that I am extremely interested in a few things but at least a little interested in nearly everything. Here I will attempt to give you a better picture of my interests by pointing to the brightest stars in the sky.

In much of what interests me, complexity and generalization are present. I like to think about complexity in two directions: (1) generation and (2) compression. I am enchanted by systems with emergent complexity, such as Lindenmayer systems or cellular automata, where complex results or histories unfold from relatively simple rules. It is this emergent complexity that I describe as generation, but its analog of compression is equally interesting. In the case of generation, I have the freedom to play with rules to find what complex thing will result. With the case of compression, I enjoy the puzzle of finding such rules that successfully or sufficiently compress the phenomenon. My interests in such topics as mathematical modeling, machine learning, and multivariate data analysis stem from trying to search for the rules that best compress complex systems. In the cases of reinforcement learning and generative adversarial learning there is a particularly enjoyable interplay between generating and compressing complexity. This description of complexity is somewhat hollow without giving some regard to what I consider to be complex.

I don't know what "complexity" is in terms of a single, unified, idea. However, I find myself engaged in different subjects that have their own operational definitions and measurements of complexity. In lieu of a definition, I'll describe those that I have so-far discovered to be fascinating.

The first kind is what I call "complexity as combinatorics", where one may find that to describe a system requires the consideration of a combinatorial explosion of possibilities. This includes Graham's colouring problem that results in Graham's Number (G_{64}) as an upper bound, or the game of (inf-embeddable) trees that results in the surprisingly huge TREE(3).

The second kind is readily related to the first, which I refer to as "complexity as difficulty", where the difficulty of a problem gives some description of its complexity. The examples of this that are most familiar to computer scientists are the time complexity and space complexity of an algorithm as a description of how efficiently a problem is solved. I've often found the results of best/average/worst case analysis useful in problem-solving, but it has also helped me appreciate the importance of the P vs NP problem in real-world problems such as protein folding and scheduling.

The third kind of complexity I think of as "complexity as chaos", where the history or outcome of a system dramatically changes with small changes in the initial (boundary) conditions of that system, or in terms of a small change in some generic parameter. My three favourite examples of chaos are (1) turbulence in fluid dynamics, where solutions to the Navier-Stokes equations are chaotic, (2) the logistic map whose single parameter dictates whether the behaviour of $x_{n+1} = rx_n(1-x_n)$ is 'simple' or chaotic, and (3) Lorenz's strange attractor where a curve through a parametric space seems to loop around two points forever without ever intersecting itself.

The fourth kind of complexity is "complexity as minimally sufficient representation", where we try to represent (compress) something as simply as possible without any loss of information. In practical situations such as image formats, I consider whether to use lossless compressions such as PNG or lossy compressions such as JPEG. Less practically, although it gives me an appreciation of code golf, is that of Kolmogorov complexity in which our goal is to write the shortest computer program in order to produce a result. In many contexts, I see such a result as a description of a system and the minimal lossless compression as its complexity.

The fifth kind of complexity, I think of as "complexity as information entropy" or "complexity as uncertainty". This kind is deeply related to the fourth kind, but its independent interpretation has often warranted special consideration in how I see stochastic systems. Like many biochemistry students, my first introduction to entropy was couched in the language of statistical thermodynamics and presented with domain-specific derivations. Over time, I've come to see information theory as both mathematically and historically more fundamental than physics in understanding the concept of entropy. Some of my favourite equations in this area include joint entropy for describing the uncertainty in joint distributions, mutual information for measuring association between variables without assumptions of linear relationships, and the minimization of cross-entropy as a loss function in order to indirectly minimize the Kullback-Leibler divergence between observed and predicted distributions.

Finally, the sixth kind of complexity that I like to think about is "emergent complexity", which is not a measure of complexity itself but rather a classification of those systems that are both complex and described/generated by relatively simple rules. As mentioned above, this includes L-systems, and cellular automata, but I've often associated it with multiple compositions of functions/maps to system states. This association has led me to iterated function systems that generate patterns like the Barnsley Fern or the divergence maps of recurrence relations such as those that visualize the Mandelbrot and Julia sets.

So that's "complexity" as far as I've understood it, but where does generalization come into this? Dabbling in many subjects, I find that every discipline solves complex problems in different ways. However, mathematics and logic have shown me that seemingly disparate topics can be brought together through abstraction, and those abstractions are often further generalized into even deeper abstractions. For example, food webs, nutrient cycles, reproductive cycles, social networks, the internet, mind maps, molecular structures, metabolic pathways, traffic, game theory, and even recreational games like snakes and ladders have all helped open my eyes to graph theory (graphs, digraphs, multigraphs), and graph theory itself helped me conquor the formalisms to understand hypergraph theory. My maxim is that for whatever concepts I have learned, they will always have generalizations that provide more mountains to climb. Furthermore, it is generalizations that have given me the framework to both generate and compress complexity, and computers have become my interface of choice between such generalizations and the complexity problems that interest me.

2 Experience

2.1 Coding and Computer Science

In 2014 I was working as a research technician at University of Saskatchewan, and I noticed that it was taking hours for the graduate students to collate their data from the outfile files of analytical instruments. I was able to learn enough Python to code a script that collated the required data, reducing the process down from hours to less than a second. I have gratitude toward my brother, a succesful software engineer working for a telecommunications company, for persuading me to try coding to overcome this problem. During my time working for a research lab, I used Python more and more until I was given the opportunity by my employer to learn about bioinformatics from one of our post docs. By the time our resident bioinformatician moved onto other opportunities, I had taken on all development of bioinformatics pipelines and significant data processing in our lab group. My employer had me attend seminars on OpenMP, giving me my first exposure to C++ and Fortran, and through Compute Canada led to me managing and processing data on Westgrid's clusters. I'm a believer in passing on knowledge, and was able to do so through giving a lecture to our graduate students on the bioinformatics typically used in their area of ecotoxicology as well as giving one-on-one training to both graduate students and post docs to facilitate their research goals.

Throughout my undergraduate degree, I made extensive use of LaTeX to prepare documents including essays, technical reports, and presentations (Beamer). For a project in Proteins and Enzymology, I wrote a Python script to automatically colour amino acids by residue type through PyMol's built-in Python interpreter. In my undergraduate thesis, I made use of R and Python to process transcriptomic data on over 5000 genes across multiple timepoints, gave a LaTeX /Graphviz presentation to the biochemistry department on my findings.

I've learned to have a basic facility with parsing HTML (Beautiful Soup) for webscraping (Requests, Selenium), coding in Markdown for easy formatting, and I've been quick to make use of parsers for other file formats including JSON and XML. I've also learned some foundational skills of SQL, including managing tables and views, and utilizing queries and functions.

While coding is certainly a part of computer science, and helped me understand aspects of computer science, I've found the wider subject interesting and perspective-changing. A book that heavily influenced me was *Algorithms to Live By: The Computer Science of Human Decisions* because it inspired me with examples from optimal stopping theory, game theory, and multi-armed bandit problems.

Whether it is algorithms like QuickSort, Dijkstra, A*, or abstract data structures like stacks, queues, arrays, or linked lists, I've found that computer science is a treasure trove of problem-solving paradigms and techniques.

2.2 Mathematics

I've taken undergraduate courses in discrete mathematics, linear algebra, and calculus, however my self-directed learning has gone well beyond those courses. Over the last 8 years, various resources available outside of school have helped me understand mathematics.

My years at University of Saskatchewan was a time when I began to delve into mathematics at a faster pace. I was introduced to many concepts including structural equation modeling, the general linear model, the generalized linear model, generalized additive models, principal component analysis, numerous inferential statistics, and concepts in Bayesian probability. My bioinformatics work deepened my understanding of matrices, distance functions, and clustering.

Since leaving my position at University of Saskatchewan, my statistical knowledge of estimation has expanded and my interest in statistical machine learning has taken flight. I've spent an enormous amount of time committed to a breadth-first exploration of machine learning models that I've trained and cross validated on publically available data sets. These models to date have included logistic regression, support vector machines, discriminant functions, kernel regression, nearest neighbors, Naive Bayes classifiers, decision trees, random forests, regression trees, multilayer perceptrons, convolutional neural networks, residual neural networks, recurrent neural networks (including LSTM), and nearly a dozen clustering algorithms. In training these models I've practiced different methods of regularization (Lasso, Ridge, Elastic Net), visualization (t-SNE), and feature engineering (dimensionality reduction, one-hot encoding). Applying machine learning techniques on real datasets has in turn given me a tangible intuition of concepts in statistical learning including the bias-variance tradeoff, Cramer-Rao bound, and hill climbing algorithms.

YouTube, while not an academically rigorous source, has inspired me to dig into real mathematical subjects thanks to channels like Vsauce showing me different types of infinity, Numberphile showing me the four colour theorem, Veritasium showing me applications of the logistic map, or 3Blue1Brown showing me Quaternions. Although not formal or rigorous, I found 3Blue1Brown's series on artificial neural networks to be an intuitive introduction to how deep learning can work. These YouTube channels (and others) taught me something notoriously difficult to teach: curiousity about mathematics, and that curiousity has led me to subjects I could not even imagine ten years ago.

I've also learned mathematical concepts from resources such as Khan Academy for its practice problems, Brilliant.org for challenging puzzles, and Project Euler for merging math puzzles with coding. Academic resources, especially opencourseware from MIT and Stanford have been instrumental in improving my knowledge of computer science, mathematics, statistics, and artificial intelligence.

I've sampled textbook and journal article readings that included subjects such as mathematical statistics, machine learning, linear algebra, multivariable calculus, non-linear mathematics, applied mathematics, probabiliy, and graph theory in such a way that I have a broad understanding of the kinds of problems tackled in each field of study.

One book that was particulary influential on me was *How to Prove It: A Structured Approach* by Daniel J. Velleman because it exposed rigorous mathematical proof approaches including contrapositive, mathematical induction, and contradiction.

I am personally grateful for the many hours that Brian Schaan of the University of Northern British Columbia volunteered to introducing me to mathematical subjects that were beyond the scope of my classes, including group theory, equivalence classes, metric spaces, and topology.