How will we do mathematics in 2030 ?

Michael R. Douglas

CMSA / Harvard University and Stony Brook University

Western Hemisphere Colloquium on Geometry and Physics,
Oct 17, 2022

Abstract

We make the case that over the coming decade, computational technology and computer assisted reasoning will become far more widely used in the mathematical sciences.

Extended Abstract

We make the case that over the coming decade, computer assisted reasoning will become far more widely used in the mathematical sciences. This includes interactive and automatic theorem verification, symbolic algebra, and emerging technologies such as formal knowledge repositories, semantic search and intelligent textbooks. After a short review of the state of the art, we survey directions where we expect progress, such as mathematical search and formal abstracts, developments in computational mathematics, integration of computation into textbooks, and organizing and verifying large calculations and proofs. For each we try to identify the barriers and potential solutions.

Computers and the internet have had a large and growing impact on many human activities. Many believe that machine learning and AI will dramatically expand this impact.

In this talk we focus on the mathematical sciences, broadly defined to include not just pure and applied mathematics and statistics, but also much of computer science, and the theoretical and computational branches of physics, chemistry and biology.

Our goal will be to identify opportunities for significant advances in how we discover, communicate and teach knowledge in these fields, which are feasible over the coming decade.

We will do this by identifying lines of research which address clear needs, which are being pursued now, which have interesting results and are making progress, and then extrapolating this progress.

Computers and the internet have had a large and growing impact on many human activities. Many believe that machine learning and AI will dramatically expand this impact.

In this talk we focus on the mathematical sciences, broadly defined to include not just pure and applied mathematics and statistics, but also much of computer science, and the theoretical and computational branches of physics, chemistry and biology.

Our goal will be to identify opportunities for significant advances in how we discover, communicate and teach knowledge in these fields, which are feasible over the coming decade.

We will do this by identifying lines of research which address clear needs, which are being pursued now, which have interesting results and are making progress, and then extrapolating this progress.

Computers and the internet have had a large and growing impact on many human activities. Many believe that machine learning and AI will dramatically expand this impact.

In this talk we focus on the mathematical sciences, broadly defined to include not just pure and applied mathematics and statistics, but also much of computer science, and the theoretical and computational branches of physics, chemistry and biology.

Our goal will be to identify opportunities for significant advances in how we discover, communicate and teach knowledge in these fields, which are feasible over the coming decade.

We will do this by identifying lines of research which address clear needs, which are being pursued now, which have interesting results and are making progress, and then extrapolating this progress.

Computers and the internet have had a large and growing impact on many human activities. Many believe that machine learning and AI will dramatically expand this impact.

In this talk we focus on the mathematical sciences, broadly defined to include not just pure and applied mathematics and statistics, but also much of computer science, and the theoretical and computational branches of physics, chemistry and biology.

Our goal will be to identify opportunities for significant advances in how we discover, communicate and teach knowledge in these fields, which are feasible over the coming decade.

We will do this by identifying lines of research which address clear needs, which are being pursued now, which have interesting results and are making progress, and then extrapolating this progress.

The first electronic computers were built at the end of World War II, and they were first used for numerical calculations in scientific and military applications. Soon the pioneers of computer science saw their far broader potential:

- Information storage and retrieval (Vannevar Bush)
- Control systems (many people, let's say Norbert Wiener)
- Artificial intelligence (Newell and Simon, McCarthy, Minsky)
- Natural computing (Rosenblatt, Ulam and von Neumann, Holland)

In mathematical research, the growing availability of computation led to a steadily growing interest in numerical methods and simulation. This has had a huge impact on the mathematical sciences.

AI researchers proposed visionary ideas such as automated theorem proving. Early attempts were based on tree search, which in general requires exponential time and by itself can only solve small problems. But as part of their research, they developed broadly useful technologies for symbol manipulation, leading for example to symbolic algebra systems.

The first electronic computers were built at the end of World War II, and they were first used for numerical calculations in scientific and military applications. Soon the pioneers of computer science saw their far broader potential:

- Information storage and retrieval (Vannevar Bush)
- Control systems (many people, let's say Norbert Wiener)
- Artificial intelligence (Newell and Simon, McCarthy, Minsky)
- Natural computing (Rosenblatt, Ulam and von Neumann, Holland)

In mathematical research, the growing availability of computation led to a steadily growing interest in numerical methods and simulation. This has had a huge impact on the mathematical sciences.

AI researchers proposed visionary ideas such as automated theorem proving. Early attempts were based on tree search, which in general requires exponential time and by itself can only solve small problems. But as part of their research, they developed broadly useful technologies for symbol manipulation, leading for example to symbolic algebra systems.

The first electronic computers were built at the end of World War II, and they were first used for numerical calculations in scientific and military applications. Soon the pioneers of computer science saw their far broader potential:

- Information storage and retrieval (Vannevar Bush)
- Control systems (many people, let's say Norbert Wiener)
- Artificial intelligence (Newell and Simon, McCarthy, Minsky)
- Natural computing (Rosenblatt, Ulam and von Neumann, Holland)

In mathematical research, the growing availability of computation led to a steadily growing interest in numerical methods and simulation. This has had a huge impact on the mathematical sciences.

AI researchers proposed visionary ideas such as automated theorem proving. Early attempts were based on tree search, which in general requires exponential time and by itself can only solve small problems. But as part of their research, they developed broadly useful technologies for symbol manipulation, leading for example to symbolic algebra systems.

The next technological leap was the Internet. This dramatically sped up the dissemination of results. In my own field of theoretical physics, we went from paper preprints in the 80's to the arXiv in the early 90's.

The "second superstring revolution" was begun in 1993 by Ashoke Sen, working almost alone in India. Before the internet, important discoveries were quickly taken over by groups working in a few dominant centers. But with the internet, Sen and the many other researchers spread around the world could stay competitive. Unlike the "first superstring revolution" in the 80's, this time there was no single dominant center of research.

Arguably the greatest success story of this type is Wikipedia. In the past, large collaborative knowledge projects such as encyclopedias required a great deal of organization, and centralized control of the editing process. While Wikipedia still has editors and a hierarchical organization, it requires far fewer editors than anyone predicted. Wikipedia is a valuable resource for mathematical scientists, but by no means a panacea.

The next technological leap was the Internet. This dramatically sped up the dissemination of results. In my own field of theoretical physics, we went from paper preprints in the 80's to the arXiv in the early 90's.

The "second superstring revolution" was begun in 1993 by Ashoke Sen, working almost alone in India. Before the internet, important discoveries were quickly taken over by groups working in a few dominant centers. But with the internet, Sen and the many other researchers spread around the world could stay competitive. Unlike the "first superstring revolution" in the 80's, this time there was no single dominant center of research.

Arguably the greatest success story of this type is Wikipedia. In the past, large collaborative knowledge projects such as encyclopedias required a great deal of organization, and centralized control of the editing process. While Wikipedia still has editors and a hierarchical organization, it requires far fewer editors than anyone predicted. Wikipedia is a valuable resource for mathematical scientists, but by no means a panacea.

The next technological leap was the Internet. This dramatically sped up the dissemination of results. In my own field of theoretical physics, we went from paper preprints in the 80's to the arXiv in the early 90's.

The "second superstring revolution" was begun in 1993 by Ashoke Sen, working almost alone in India. Before the internet, important discoveries were quickly taken over by groups working in a few dominant centers. But with the internet, Sen and the many other researchers spread around the world could stay competitive. Unlike the "first superstring revolution" in the 80's, this time there was no single dominant center of research.

Arguably the greatest success story of this type is Wikipedia. In the past, large collaborative knowledge projects such as encyclopedias required a great deal of organization, and centralized control of the editing process. While Wikipedia still has editors and a hierarchical organization, it requires far fewer editors than anyone predicted. Wikipedia is a valuable resource for mathematical scientists, but by no means a panacea.

Many people would agree that we are in the midst of a third technological leap, powered by **machine learning**. ML has led to dramatic improvements in the ability of computers to recognize and classify patterns, to translate between languages, to play games of strategy, to extract information from documents, and to carry out tasks without requiring explicit programming. Its scope is still expanding.

ML is already a core technology for firms like Google, Apple, Facebook *etc.* and they have established groups whose total research staff numbers in the thousands. ML is gradually being adopted by businesses of all sizes, academic research groups, government *etc.*

How can ML be used by mathematical scientists? Of course ML is itself a mathematical concept and is studied by statisticians, applied mathematicians and other mathematical scientists. But it can also be used to augment the human capability to do more general research, by helping to recognize patterns, by finding and organizing relevant data, documents and code, by helping to write and verify code and mathematical proofs, in education, and in other ways.

Many people would agree that we are in the midst of a third technological leap, powered by **machine learning**. ML has led to dramatic improvements in the ability of computers to recognize and classify patterns, to translate between languages, to play games of strategy, to extract information from documents, and to carry out tasks without requiring explicit programming. Its scope is still expanding.

ML is already a core technology for firms like Google, Apple, Facebook *etc.* and they have established groups whose total research staff numbers in the thousands. ML is gradually being adopted by businesses of all sizes, academic research groups, government *etc.*

How can ML be used by mathematical scientists? Of course ML is itself a mathematical concept and is studied by statisticians, applied mathematicians and other mathematical scientists. But it can also be used to augment the human capability to do more general research, by helping to recognize patterns, by finding and organizing relevant data, documents and code, by helping to write and verify code and mathematical proofs, in education, and in other ways.

Many people would agree that we are in the midst of a third technological leap, powered by **machine learning**. ML has led to dramatic improvements in the ability of computers to recognize and classify patterns, to translate between languages, to play games of strategy, to extract information from documents, and to carry out tasks without requiring explicit programming. Its scope is still expanding.

ML is already a core technology for firms like Google, Apple, Facebook *etc.* and they have established groups whose total research staff numbers in the thousands. ML is gradually being adopted by businesses of all sizes, academic research groups, government *etc.*

How can ML be used by mathematical scientists? Of course ML is itself a mathematical concept and is studied by statisticians, applied mathematicians and other mathematical scientists. But it can also be used to augment the human capability to do more general research, by helping to recognize patterns, by finding and organizing relevant data, documents and code, by helping to write and verify code and mathematical proofs, in education, and in other ways.

Computational mathematics

Many of the great mathematicians – Euler, Gauss and Ramanujan come to mind – were renowned for the calculational abilities, which were the basis for many mathematical discoveries.
This tradition continues and has been enhanced by the use of computers. Leaving aside computer-aided proofs of earlier conjectures, a few discoveries which were made this way include

- Many properties of chaos in dynamical systems.
- Most of the original constructions of the sporadic finite groups.
- The Birch Swinnerton-Dyer conjecture, based on calculations done on the EDSAC-2 computer at Cambridge.

Numerical experiment is now a central part of number theory, see for example the web site of the Simons Collaboration on Arithmetic Geometry, Number Theory, and Computation.
Still, continuing any specific program eventually runs into exponentially large computing problems, because of the curse of dimensionality, the existence of NP hard problems, *etc.* .

So what are **new** ideas and tools which will lead to progress by 2030 ?

- Faster computers, more storage, better infrastructure
- Satisfiability modulo theories, SAT+CAS, ...
- Neural networks – the subject of many talks at this workshop
- Advances in statistics

The first of these, while not involving deep concepts, can still make a big difference to research. As an example from physics, there was a large effort starting in the 80's to do numerical simulations of lattice gauge theory, to compute masses of hadrons *ab initio*. This problem was essentially solved in the 2000's, mostly thanks to cheap supercomputers (clusters with GPUs).

If we were within a factor of 1000 (time, space, ...) of solving a problem in 2019, it will probably be solved using the same methods by 2030.

SAT, SMT and SAT+CAS

SAT solvers: solve problems in propositional logic, given as a list of clauses.
Math example: the Boolean Pythagorean triple problem, split $\{1...N\}$ into two subsets such that neither subset contains a triple $a^2 + b^2 = c^2$.
Possible for $N = 7824$ and not for $N = 7825$, as shown by Heule, Kullmann, and Marek in 2016 (producing "the world's longest proof").

In this problem the Boolean variables are the assignments of each number to a subset, and the logical clauses are easy to derive. Many problems involve more algebra, or reasoning in other domains. An SMT (Satisfiability Modulo Theories) solver combines some other decision algorithm with a SAT solver. Often the SAT solver can reach its conclusions without evaluating many of the constraints, so this will be much faster. The "SAT+CAS" variant combines a SAT solver with a general computer algebra system, which checks proposed solutions and can return "conflict clauses" to the SAT solver.

SAT, SMT and SAT+CAS

SAT solvers: solve problems in propositional logic, given as a list of clauses.
Math example: the Boolean Pythagorean triple problem, split $\{1...N\}$ into two subsets such that neither subset contains a triple $a^2 + b^2 = c^2$. Possible for $N = 7824$ and not for $N = 7825$, as shown by Heule, Kullmann, and Marek in 2016 (producing "the world's longest proof").

In this problem the Boolean variables are the assignments of each number to a subset, and the logical clauses are easy to derive. Many problems involve more algebra, or reasoning in other domains. An SMT (Satisfiability Modulo Theories) solver combines some other decision algorithm with a SAT solver. Often the SAT solver can reach its conclusions without evaluating many of the constraints, so this will be much faster. The "SAT+CAS" variant combines a SAT solver with a general computer algebra system, which checks proposed solutions and can return "conflict clauses" to the SAT solver.

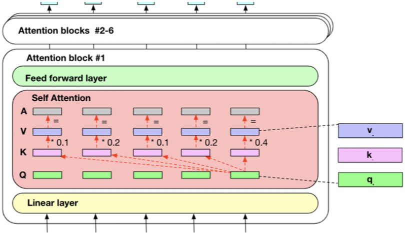
Problems suitable for this approach tend to be those in which the search space has a natural Boolean encoding. There are many examples in `https://arxiv.org/abs/1907.04408` (the MathCheck project):

- The Williamson conjecture: find four symmetric $n \times n$ matrices with ± 1 entries such that

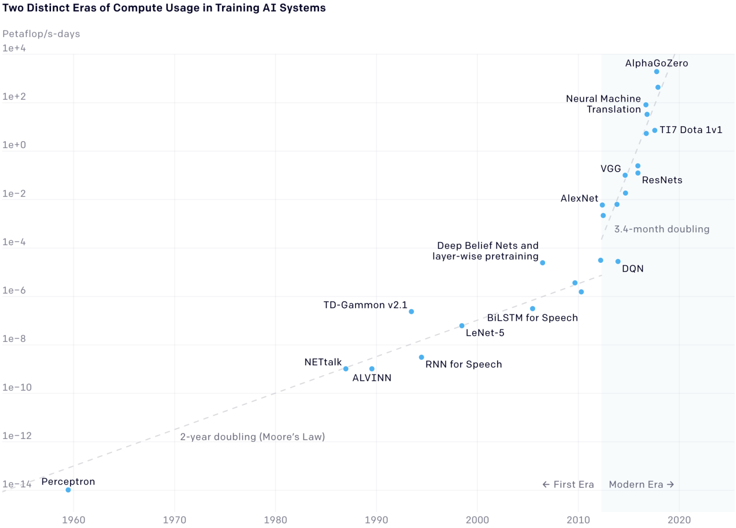
$$A^2 + B^2 + C^2 + D^2 = 4n \cdot \mathrm{id}. \tag{1}$$

  These can be arranged into a $4n \times 4n$ Hadamard matrix satisfying $HH^t = 4n \cdot$ id. Strangely enough, these exist for all $n < 35$, and all even $n \leq 70$, but not for $n = 35$.
- Golay pairs: polynomials f, g with coefficients from $\{1, i, -1, -i\}$ such that $|f(z)|^2 + |g(z)|^2$ is constant on the unit circle.
- 3×3 matrix multiplication using fewer than 27 scalar multiplications. One can find many ways using 23 and so far, none using 22.

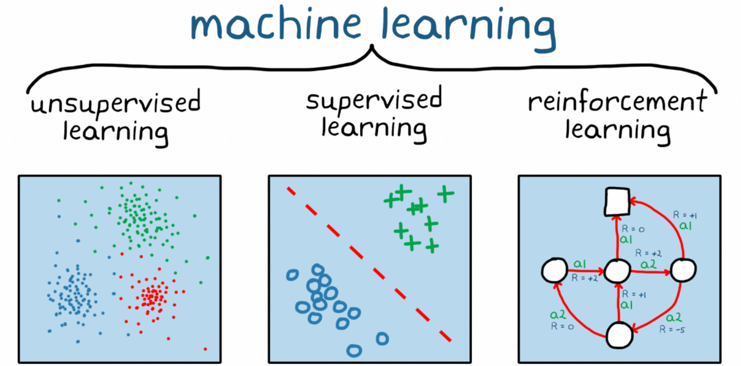
Machine Learning and Neural Networks



AlphaFold: a solution to a 50-year-old grand challenge in biology


Two Distinct Eras of Compute Usage in Training AI Systems
Petaflop/s-days
1e+4
1e+2
1e+0
1e−2
1e−4
1e−6
1e−8
1e−10
1e−12
1e−14
AlphaGoZero
Neural Machine Translation
TI7 Dota 1v1
VGG
ResNets
AlexNet
3.4-month doubling
Deep Belief Nets and layer-wise pretraining
DQN
TD-Gammon v2.1
BiLSTM for Speech
LeNet-5
NETtalk
ALVINN
RNN for Speech
2-year doubling (Moore's Law)
Perceptron
← First Era
Modern Era →
1960
1970
1980
1990
2000
2010
2020

Generalities on machine learning



Three standard paradigms: supervised learning (function fitting), reinforcement learning (usually, fitting a Markov decision process), and self-supervised learning (fitting a conditional probability distribution).

- All ML methods involve fitting parameterized functions to data, almost always by optimizing an objective function, most commonly by stochastic gradient descent.
- In “deep learning” the standard parameterized function is a feedforward neural network. These fits are powered by techniques to efficiently evaluate the models and compute gradients with respect to parameters (GPUs, backpropagation).
- Current methods require huge datasets. Performance often goes as a low power (e.g. $N^{1/10}$) of the dataset size and number of parameters.
- Supervised learning requires labeled data. Large curated datasets (e.g. CIFAR-10, Imagenet) led to major advances. But these are hard to get.
- Self-supervised learning of masked word prediction led to the tremendous advances in large language models (Bert, GPT-3, PaLM).
- Reinforcement learning for game self-play can generate arbitrarily large quantities of synthetic data, as was done for AlphaZero .

Synthetic datasets and model-informed learning

- Statistics and standard ML are based on general purpose models. By contrast, physics has fundamental models which accurately describe the dynamics of a system: Newton's equations of celestial mechanics, the Schrödinger equation of atomic physics, etc.. Model-informed ML methods use the particularities of these models to do inference.
- One general approach is to simulate the physical equations starting from diverse initial conditions to get a set of trajectories: a synthetic dataset. Model features can also use known properties of solutions.
- Mathematical objects come in sets or families which can be classified: Riemann surfaces and higher dimensional manifolds, the elliptic curves of number theory, knots in three dimensions, etc. These are platonic datasets, *i.e.* synthetic datasets depending on few arbitrary choices (one might need to choose a bound on the size of the objects).

A mathematical dataset - knots

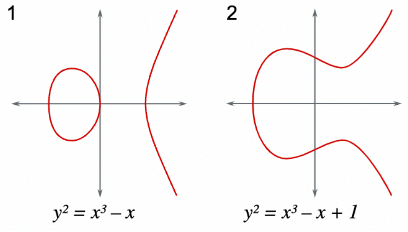
One mathematically natural (or “platonic”) and arbitrarily large dataset is a table of knots. A table of all knots with up to 16 crossings (over a million knots) with their corresponding knot invariants has been computed.

z: **Knot**	$X(z)$: **Geometric invariants**				$Y(z)$: **Algebraic invariants**		
	Volume	Chern–Simons	Meridional translation	...	Signature	Jones polynomial	...
	2.0299	0	i	...	0	$t^{-2} - t^{-1} + 1 - t + t^2$	...
	2.8281	−0.1532	$0.7381 + 0.8831i$	...	−2	$t - t^2 + 2t^3 - t^4 + t^5 - t^6$	...
	3.1640	0.1560	$-0.7237 + 1.0160i$	...	0	$t^{-2} - t^{-1} + 2 - 2t + t^2 - t^3 + t^4$	...

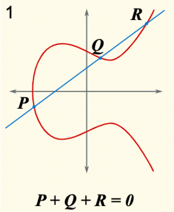
Fig. 2 | Examples of invariants for three hyperbolic knots. We hypothesized that there was a previously undiscovered relationship between the geometric and algebraic invariants.

Davies et al (2021) fit this data to predict the signature from the other invariants, and working with knot theorist Marc Lackenby proved a new relation between invariants.

A number theoretic dataset: elliptic curves



Solutions of a cubic equation - one parameter



Group law

The group law on an elliptic curve is abelian and takes rational points to rational points. The rank of an elliptic curve is the number of generators of the infinite order part of the group. Most elliptic curves have rank 0 or 1, but examples are known with ranks 20 and higher. There are large datasets of elliptic curves, such as the LMFDB at https://www.lmfdb.org/ .

The Birch Swinnerton-Dyer conjecture: one of the first (1960's) mathematical conjectures based on machine learning

The **Birch and Swinnerton-Dyer** conjecture is one of the Millennium Prize Problems listed by the Clay Mathematics Institute. It relates the order of vanishing and the first non-zero Taylor series coefficient of the L-function associated to an elliptic curve E defined over $\mathbb{Q}$ at the central point $s = 1$ to certain arithmetic data, the BSD invariants of E.

Specifically, the BSD conjecture states that the order r of vanishing of $L(E, s)$ at $s = 1$ is equal to the rank of the Mordell-Weil group $E(\mathbb{Q})$, and that

$$\frac{1}{r!} L^{(r)}(E, 1) = \frac{\#Ш(E/\mathbb{Q}) \cdot \Omega_E \cdot \mathrm{Reg}(E/\mathbb{Q}) \cdot \prod_p c_p}{\#E(\mathbb{Q})^2_{\mathrm{tor}}}.$$

The quantities appearing in this formula are the BSD invariants of E:

- r is the rank of $E(\mathbb{Q})$ (a non-negative integer);
- $\#Ш(E/\mathbb{Q})$ is the order of the Tate-Shafarevich group of E (which is conjectured to always be finite, a positive integer);
- $\mathrm{Reg}(E/\mathbb{Q})$ is the regulator of $E/\mathbb{Q}$;
- Ω_E is the real period of $E/\mathbb{Q}$ (a positive real number);
- c_p is the Tamagawa number of E at each prime p (a positive integer which is 1 for all but at most finitely many primes);
- $E(\mathbb{Q})_{\mathrm{tor}}$ is the torsion order of $E(\mathbb{Q})$ (a positive integer).

Predicting (in)stability of an exoplanetary solar system (Tamayo *et al* 2020)

The long term stability of our Solar System was famously studied by Laplace and Lagrange (among others). Using perturbation theory one can show stability for 10000's of years but also see that it is not guaranteed over longer timescales. In fact the dynamics is chaotic and depends sensitively on the initial conditions.

In 2009 Gastineau and Lascar showed that the Solar System is **not** stable, by starting from 1000's of nearby initial conditions and simulating for billions of years. About 1% of the starting points led to collisions or ejection of a planet.

The discovery of exoplanetary systems leads us to ask the general question: given a system of N planets with masses and initial conditions, predict the timescale T of (in)stability. The Lyapunov exponent (exponential growth of perturbations) gives a lower bound but chaos need not lead to instability.

One can use ML to predict T by doing many simulations to generate a synthetic dataset. The inputs include the masses and a time series of positions and velocities, or features derived from these. The target is (a lower bound on) T.

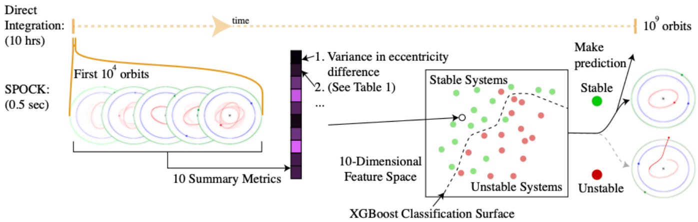
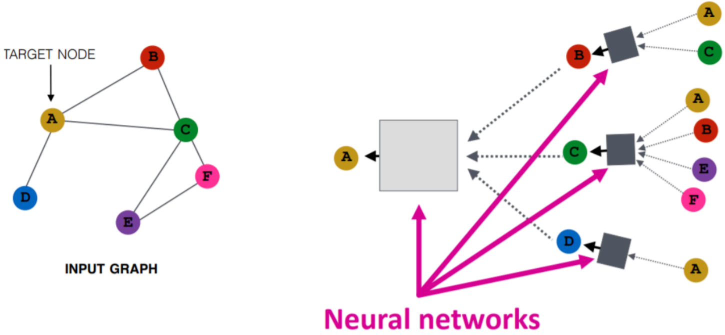


Fig. 1. A schematic illustrating how SPOCK classifies stability of a given initial configuration. The traditional approach numerically integrates the system for 10^9 orbits, requiring roughly 10 hours with current hardware. SPOCK runs a much shorter 10^4 orbit integration, and from it generates a set of 10 summary features (see Table 1). These map to a point (white circle) in a 10-dimensional space, in which we have trained an XGBoost model to classify stability. SPOCK outputs an estimated probability that the given system is stable over 10^9 orbits, up to 10^5 times faster than direct integration.

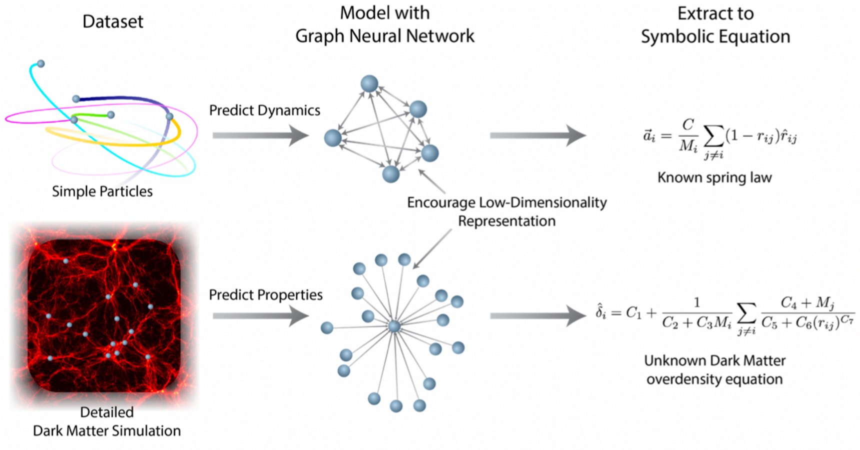
From Tamayo et al (arXiv:2007.06521) and Cranmer et al (arXiv:2101.04117)

Graphical Neural Networks



A GNN propagates data by message passing along edges, followed by aggregation (sum, max, ...) at each node, followed by a general function (FFN). This is repeated in (usually 2 or 3) layers.

Fitting modifications to the inverse square law of gravity caused by interactions with dark matter (Cranmer et al 2020)



Gravity models of trade fit by GNN (Verstyuk and Douglas 2022)

$$\mathrm{Trade}(i \leftrightarrow j) \approx \gamma \frac{\mathrm{GDP}_i \times \mathrm{GDP}_j}{\mathrm{Distance}(i,j)}$$ (Tinbergen 1962)

- CEPII Gravity Dataset includes trade, GDP, population, trade agreements (224 economies, 1948 to 2016 period, bilateral)
- Even the simplest (linear) model fits well, $R^2 \sim 0.4$
- Much work on identifying other “distance” factors, e.g. trade agreements
- Higher order interactions, for example competitive advantage. Structural models with market access factors (upweight a country’s access by a weighted average of its total imports or exports) can get $R^2 \sim 0.5$
- Verstyuk and Douglas (2022) developed a graphical neural network model with general higher order interactions. Fitting the data followed by symbolic regression leads to terms similar to market access factors and $R^2 \sim 0.6$

Table 5: Interpretation by Symbolic Regression with CES restriction, best Flexible model

Model specification	R^2 IS	R^2 OOS
$\ln X_{ij} = \ln G + f_i^M + f_j^S + \ln \phi_{ij} + \xi_{ij}^X$	0.66	0.67

Composition

Estimated approximations		Symbolic form		
Function, $\hat{f}(\boldsymbol{z}, \boldsymbol{\theta})$	Arguments, $\boldsymbol{z}$	Function, $\tilde{f}(\boldsymbol{z}, \boldsymbol{\vartheta})$	R^2 IS	MSE IS
$\hat{f}^M(\cdot)$	$X_i, \sum_k \hat{f}^M_{\phi Y}(\cdot), \sum_k \hat{f}^M_{\phi X}(\cdot)$	$\vartheta_1 \ln(\frac{\vartheta_{10}+\vartheta_{11}z_1}{\vartheta_{20}+\vartheta_{21}z_2}) + \vartheta_3 z_3$	0.96	0.15
$\hat{f}^S(\cdot)$	$Y_j, \sum_k \hat{f}^S_{\phi X}(\cdot), \sum_k \hat{f}^S_{\phi Y}(\cdot)$	$\vartheta_1 \ln(\frac{\vartheta_{10}+\vartheta_{11}z_1}{\vartheta_{30}+\vartheta_{31}z_3}) + \vartheta_2 z_2$	0.95	0.21
$\ln \hat{f}^M_{\phi X}(\cdot), \ln \hat{f}^S_{\phi X}(\cdot)$	$\hat{\phi}_{kj}, X_k$	$\ln(\vartheta_0(\vartheta_{10}z_1^{\vartheta_{11}} + \vartheta_{20}z_2^{\vartheta_{21}})^{\vartheta_1})$	0.90	664.83
$\ln \hat{f}^M_{\phi Y}(\cdot), \ln \hat{f}^S_{\phi Y}(\cdot)$	$\hat{\phi}_{ik}, Y_k$	$\ln(\vartheta_0(\vartheta_{10}z_1^{\vartheta_{11}} + \vartheta_{20}z_2^{\vartheta_{21}})^{\vartheta_1})$	0.96	118.91

Descriptive statistics		Numeric form
Mean [St.Dev.]	Mean [St.Dev.]	Function, $\tilde{f}(\boldsymbol{z}, \hat{\boldsymbol{\vartheta}})$
−1.39 [1.97]	150.47, 16.95, 7.19 [790.89, 13.40, 7.22]	$1.0055 \ln(\frac{0.0618+1.9559z_1}{11.7018+0.9271z_2}) + 0.4388z_3$
−2.05 [2.16]	149.71, 3.87, 19.61 [771.65, 4.75, 14.17]	$0.9763 \ln(\frac{0.7828+1.5192z_1}{7.5665+0.1926z_3}) - 0.0435z_2$
−30.53, −30.53 [80.24, 80.24]	0.06, 355.45 [0.18, 1259.07]	$\ln(0.1986(6.4708z_1^{15.0746} + 0.1988z_2^{0.0162})^{1.1231})$
−37.17, −37.17 [57.90, 57.90]	0.06, 345.01 [0.18, 1222.06]	$\ln(0.2979(0.0050z_1^{1.4748} + 0.2335z_2^{0.8234})^{0.9847})$

Notes: Number of observations for each component is 515345 in the training sample and 54355 in the testing sample (the latter sample is not us separately for outer and inner functions). Number of epochs for each component is 1000. Bilateral accessibility term $\ln \phi_{ij}$ is taken as an exogeno input. Parameter vectors $\boldsymbol{\theta}$ and $\boldsymbol{\vartheta}$ as well as input variable vectors $\mathbf{z}$ are component-specific. Time subscripts are omitted throughout. See text f additional details on symbolic regression and CES specification.

Advances in statistics

Statistics is hardly new, but one could argue that with less need for computational efficiency, one is more free to use general approaches based on simple concepts:

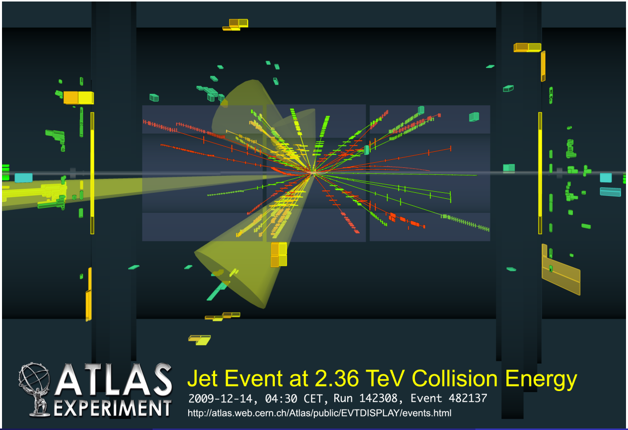
- Generative models: give entire probability distribution of data.
- Bayesian statistics: turn around model $\Rightarrow$ data to infer P(model).
- Information theory: KL divergence, variational methods
- Distances between observations or measures: Wasserstein distance and optimal transport, ...

Given two measures μ and ν on a metric space M, the p'th Wasserstein distance between them is

$$W_p(\mu,\nu) = \left(\inf_{\gamma\in\Gamma(\mu,\nu)} \int_{M\times M} d(x,y)^p d\gamma(x,y) \right)^{1/p} \quad (2)$$

where $\int_{M_1} d\gamma = \mu$, resp. M_2 and ν.

An interesting trend in physics and other hard sciences is to replace hand-crafted concepts and tools for data analysis, with more general statistical tools. An example from particle physics is Thaler *et al*, `http://arxiv.org/abs/1902.02346`, "The Metric Space of Collider Events." They use Wasserstein W_1 as a distance between collider events, considered as energy distributions μ, ν.



Probabilistic models in number theory

Interestingly, pure mathematicians also seem to be more interested in probabilistic approaches. This is not a new idea – one of the classic approaches to the theory of prime numbers is to make models in which the probability of a given number being divisible by $2, 3, \ldots$ are approximated as independent. But the availability of large databases allows making and testing more sophisticated models.

A recent example is the probabilistic model of the arithmetic of elliptic curves developed by Bhargava *et al* `arXiv:1304.3971` . To give just a taste, we are discussing the solutions to a cubic equation like $y^2 = x^3 - 2x + 2$ in rational numbers, for example $x = 1, y = \pm 1$. In fact this curve has infinitely many rational solutions, but they can all be generated from a (well chosen) pair of solutions – it is rank 1.

Curves are known with rank 19 and higher, and until recently it was believed that arbitrarily high rank should exist. But the probabilistic model suggests that the rank is ≤ 21 for all but finitely many curves.

Probabilistic models in number theory

Interestingly, pure mathematicians also seem to be more interested in probabilistic approaches. This is not a new idea – one of the classic approaches to the theory of prime numbers is to make models in which the probability of a given number being divisible by $2, 3, \ldots$ are approximated as independent. But the availability of large databases allows making and testing more sophisticated models.

A recent example is the probabilistic model of the arithmetic of elliptic curves developed by Bhargava *et al* `arXiv:1304.3971` . To give just a taste, we are discussing the solutions to a cubic equation like $y^2 = x^3 - 2x + 2$ in rational numbers, for example $x = 1, y = \pm 1$. In fact this curve has infinitely many rational solutions, but they can all be generated from a (well chosen) pair of solutions – it is rank 1.

Curves are known with rank 19 and higher, and until recently it was believed that arbitrarily high rank should exist. But the probabilistic model suggests that the rank is ≤ 21 for all but finitely many curves.

Probabilistic models in number theory

Interestingly, pure mathematicians also seem to be more interested in probabilistic approaches. This is not a new idea – one of the classic approaches to the theory of prime numbers is to make models in which the probability of a given number being divisible by $2, 3, \ldots$ are approximated as independent. But the availability of large databases allows making and testing more sophisticated models.

A recent example is the probabilistic model of the arithmetic of elliptic curves developed by Bhargava *et al* `arXiv:1304.3971` . To give just a taste, we are discussing the solutions to a cubic equation like $y^2 = x^3 - 2x + 2$ in rational numbers, for example $x = 1, y = \pm 1$. In fact this curve has infinitely many rational solutions, but they can all be generated from a (well chosen) pair of solutions – it is rank 1.

Curves are known with rank 19 and higher, and until recently it was believed that arbitrarily high rank should exist. But the probabilistic model suggests that the rank is ≤ 21 for all but finitely many curves.

Interactive theorem verification

In the 50's and 60's, the pioneers of AI developed automatic theorem provers, which generate logical deductions and search for proofs of given logical statements. Concurrently, the subfield of formal methods was developed, in which computer programs were given precise semantics allowing them to be rigorously verified.

This is of great practical value, especially for programs (an airplane autopilot, a CPU floating point unit) where mistakes can be extremely expensive. Thus it has been pursued intensively for decades, the formal methods community is fairly large and well-funded, and most of the systems we cited (Isabelle and Coq, though not Mizar) have software verification as the primary application.

Interactive theorem verification

In the 50's and 60's, the pioneers of AI developed automatic theorem provers, which generate logical deductions and search for proofs of given logical statements. Concurrently, the subfield of formal methods was developed, in which computer programs were given precise semantics allowing them to be rigorously verified.

This is of great practical value, especially for programs (an airplane autopilot, a CPU floating point unit) where mistakes can be extremely expensive. Thus it has been pursued intensively for decades, the formal methods community is fairly large and well-funded, and most of the systems we cited (Isabelle and Coq, though not Mizar) have software verification as the primary application.

As a prototypical example of software verification, let us briefly discuss sorting a list into alphabetical order. In an algorithms course one learns that a list of N elements can be sorted in worst case time $N \log N$, but the algorithms (quicksort, heapsort, ...) are a bit tricky. On the other hand, logically defining the problem of sorting is not difficult. In Coq we can say

```
Definition is_a_sorting_algorithm (f: list nat → list nat) :=
  ∀ al, Permutation al (f al) ∧ sorted (f al).
Definition sorted' (al: list nat) :=
 ∀ i j, i < j < length al → nth i al 0 ≤ nth j al 0.

Inductive Permutation : list A -> list A -> Prop :=
| perm_nil: Permutation [] []
| perm_skip x l l' : Permutation l l' -> Permutation (x::l) (x::l')
| perm_swap x y l : Permutation (y::x::l) (x::y::l)
| perm_trans l l' l'' :
    Permutation l l' -> Permutation l' l'' -> Permutation l l''.
```

In `https://softwarefoundations.cis.upenn.edu/vfa-current/Sort.html` one can see formally verified proofs that runnable sorting programs satisfy this specification.

The Fundamental Theorem of Algebra

Let's look a bit at a mathematical example, the fundamental theorem of algebra. As we all know, this states that the field $\mathbb{C}$ of complex numbers is algebraically closed, in other words every nonconstant polynomial $f(z)$ has a root.

This claim can be easily formalized: in the Lean theorem proving language, we can say

$$\text{lemma exists\_root} \quad \{f : \text{polynomial}\,\mathbb{C}\} \quad (\text{hf} : 0 < \text{degree}\, f) : \exists z : \mathbb{C}, f.\text{eval}\, z = 0 := \tag{3}$$

followed by the proof.

Here is an informal proof. We start by assuming that f has no zero, to get a contradiction. (A constructive proof exists but is longer.)

1. We first show that $|f(z)|$ attains its minimum at some point z_0. A polynomial goes to infinity at infinity, so the infimum of f is contained in a closed bounded region R. Since $|f|$ is continuous, the image of R is closed and bounded, so it contains its infimum.
2. Expand around the location z_0 of the minimum by writing
$$f(z) = f(z_0) + (z - z_0)^n g(z)$$
for some polynomial $g(z)$ such that $g(z_0) \neq 0$.
3. Now, consider a small circle $z = z_0 + \delta e^{i\theta}$. If we neglect the variation of $g(z)$ and look at $F(z) = f(z_0) + (z - z_0)^n g(z_0)$, it is easy to show that z_0 cannot be a minimum of $|F(z)|$, since $(z - z_0)^n$ takes every possible phase.
4. Intuitively, we then want to choose δ small enough such that we can neglect the variation of $g(z)$, so z_0 cannot be a minimum of $|f(z)|$, a contradiction. After a bit of algebra, it turns out that $|g(z) - g(z_0)| < |g(z_0)| \forall |z - z_0| \leq \delta$ suffices.

The Lean version of this proof takes about 100 lines: here is part 1.

```
12  lemma exists_forall_abs_polynomial_eval_le (p : polynomial ℂ) :
13    ∃ x, ∀ y, (p.eval x).abs ≤ (p.eval y).abs :=
14  if hp0 : 0 < degree p
15  then let ⟨r, hr0, hr⟩ := polynomial.tendsto_infinity complex.abs hp0 ((p.eval 0).abs) in
16    let ⟨x, hx₁, hx₂⟩ := exists_forall_le_of_compact_of_continuous (λ y, (p.eval y).abs)
17      (continuous_abs.comp p.continuous_eval)
18      (closed_ball 0 r) (proper_space.compact_ball _ _)
19      (set.ne_empty_iff_exists_mem.2 ⟨0, by simp [le_of_lt hr0]⟩) in
20    ⟨x, λ y, if hy : y.abs ≤ r then hx₂ y $ by simpa [complex.dist_eq] using hy
21      else le_trans (hx₂ _ (by simp [le_of_lt hr0])) (le_of_lt (hr y (lt_of_not_ge hy)))⟩
22  else ⟨p.coeff 0, by rw [eq_C_of_degree_le_zero (le_of_not_gt hp0)]; simp⟩
```

- The statement being proved is on line 13 - it is clear and intuitive.
- The language is "computerese" - but this is a question of taste and one can display the same content in more math-friendly notations.
- Unlike the informal proof, we had to give many propositions their own names and calling conventions, which also hurts readability.
- Commands like `rw` and `simpa` are "tactics," explicit instructions to the proof verifier. These are procedural and tricky to get right.

Today's ITV systems incorporate many advances in logic, such as dependent type theory (in Coq and Lean), and many advances in computer science. But despite all this work, theorem verification is more akin to programming than to any of the traditional skills of a mathematical scientist. And the many differences with informal proofs which we just cited, while each fairly simple, add up. At present ITV is hard to learn and use.

Still, many people believe that formalization and verification is a central part of the relation between computers and mathematics. This even includes theoretical physicists and others for whom rigorous proof is not a primary goal. It is hard to get a computer to understand anything, and here is a way for it to "understand truth." So how to use this?

- Definitions are easier to write than proofs: focus on these?
- Perhaps ITVs need more reasoning methods than deductive logic? Say counterexamples, heuristics, *etc.*. Omitted in this talk.
- Will machine learning and AI help?

Today's ITV systems incorporate many advances in logic, such as dependent type theory (in Coq and Lean), and many advances in computer science. But despite all this work, theorem verification is more akin to programming than to any of the traditional skills of a mathematical scientist. And the many differences with informal proofs which we just cited, while each fairly simple, add up. At present ITV is hard to learn and use.

Still, many people believe that formalization and verification is a central part of the relation between computers and mathematics. This even includes theoretical physicists and others for whom rigorous proof is not a primary goal. It is hard to get a computer to understand anything, and here is a way for it to "understand truth." So how to use this?

- Definitions are easier to write than proofs: focus on these?
- Perhaps ITVs need more reasoning methods than deductive logic? Say counterexamples, heuristics, *etc.*. Omitted in this talk.
- Will machine learning and AI help?

Today's ITV systems incorporate many advances in logic, such as dependent type theory (in Coq and Lean), and many advances in computer science. But despite all this work, theorem verification is more akin to programming than to any of the traditional skills of a mathematical scientist. And the many differences with informal proofs which we just cited, while each fairly simple, add up. At present ITV is hard to learn and use.

Still, many people believe that formalization and verification is a central part of the relation between computers and mathematics. This even includes theoretical physicists and others for whom rigorous proof is not a primary goal. It is hard to get a computer to understand anything, and here is a way for it to "understand truth." So how to use this?

- Definitions are easier to write than proofs: focus on these?
- Perhaps ITVs need more reasoning methods than deductive logic? Say counterexamples, heuristics, *etc.*. Omitted in this talk.
- Will machine learning and AI help?

Formal Abstracts

Tom Hales at U Pittsburgh has begun a project to create an online repository of formal abstracts, meaning statements of the main results of a mathematical paper, expressed in formal terms. Proofs would not be required, but it should be possible in principle to prove every abstract true or false. This project has many parts – here are a few (based on discussions with Tom):

- A solid ITV with dependent types – Lean.
- A library of standard concepts which can be used by abstracts, probably covering all of advanced undergraduate/early graduate level mathematics. A very rough estimate of the size is about 50,000 definitions filling 10,000 pages.
- Abstracts can be written in a controlled natural language which looks like standard mathematical text.
- Interactive tools to help search for, read and write abstracts.

While ambitious, such a system could be fully operational with its library in less than five years.

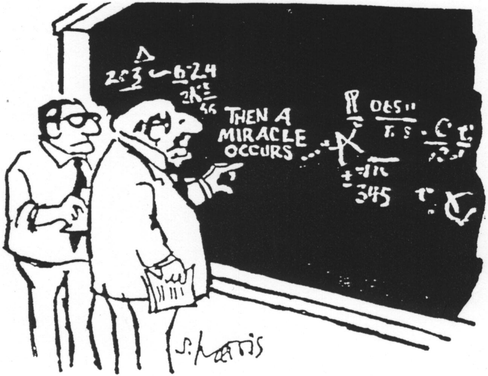
Formalization of research level mathematics

Here are some recent and current projects in formalization of research mathematics, including central results with substantial proofs, and frontier topics. See recent articles and blogs of Kevin Buzzard, and his upcoming ICM lecture, for the latest developments.

- Schemes in algebraic geometry (Buzzard *et al* arXiv:2101.02602).
- Peter Scholze's "Liquid Tensor Experiment" – formalize the proof of a key theorem about condensed abelian groups in *Lectures on Analytic Geometry* by Clausen and Scholze. Completed this summer by Johan Commelin and the Lean community.
- Perfectoid spaces (Buzzard, Commelin, Massot).
- Independence of the continuum hypothesis (Han and van Doorn, arXiv:2102.02901).
- In progress (Massot *et al*): the h-principle for open and ample first order differential relations, and its application to sphere eversion (a homotopy of embeddings $S^2 \to \mathbb{R}^3$ which exchanges inside and outside).

As for the third question, clearly AI has made transformative progress over the last decade. Let's come back to this after discussing some more applications.

AI and theorem proving



I think you should use deep learning , here in Step 2

Many groups are applying machine learning to make ITV more automatic, in AITP projects such as TacticToe (Gauthier, Kalisczyk and Urban 2017), GamePad (Huang, Dhariwal, Song and Sutskever 2018), HOList (Bansal, Loos, Rabe, Szegedy and Wilcox 2019), CoqGym (Yang and Deng, 2019), and GPT-F (Han *et al*, 2021).

In developing a proof, many choices must be made, including

- Premise selection. Out of the many known true statements, which ones should be used to make the next deduction? There could be 100's of candidates in the current context, and if we search the entire library of proved theorems, millions of candidates.
- Tactic selection. Tactics include introduction of antecedent clauses, rewriting and simplification, and other simple logical steps. Modern ITV's typically provide 40–100 tactics.

We can make an analogy between these successive choices, and the moves in a game of solitaire.

Many groups are applying machine learning to make ITV more automatic, in AITP projects such as TacticToe (Gauthier, Kalisczyk and Urban 2017), GamePad (Huang, Dhariwal, Song and Sutskever 2018), HOList (Bansal, Loos, Rabe, Szegedy and Wilcox 2019), CoqGym (Yang and Deng, 2019), and GPT-F (Han *et al*, 2021).

In developing a proof, many choices must be made, including

- Premise selection. Out of the many known true statements, which ones should be used to make the next deduction? There could be 100's of candidates in the current context, and if we search the entire library of proved theorems, millions of candidates.
- Tactic selection. Tactics include introduction of antecedent clauses, rewriting and simplification, and other simple logical steps. Modern ITV's typically provide 40–100 tactics.

We can make an analogy between these successive choices, and the moves in a game of solitaire.

A very successful approach to AI game play is reinforcement learning (RL), as famously used by AlphaGo. Its central parts are a pair of neural networks, one to choose moves and the other to evaluate the score of game positions. The original AlphaGo used a corpus of human games for initial training, and then generated games by self-play. (Another important element is Monte Carlo tree search – it turns out for Go that playing out a game many times with random moves, gives a good estimate for its score.)

Similarly, the AITP systems use a corpus of proven theorems as training data. As the verifier works through a theorem, each step of premise and tactic selection is saved, along with a summary of the state just before the choice is made. One can then use these pairs (state, selection) to train networks to do premise and tactic selection.

The better developed systems (Coq, HOL) have large libraries with 30,000–70,000 (short) theorems. This is enough training data to achieve success rates in proving similar theorems (a held-out testing set) of around 75% (it was 50% two years ago).

A very successful approach to AI game play is reinforcement learning (RL), as famously used by AlphaGo. Its central parts are a pair of neural networks, one to choose moves and the other to evaluate the score of game positions. The original AlphaGo used a corpus of human games for initial training, and then generated games by self-play. (Another important element is Monte Carlo tree search – it turns out for Go that playing out a game many times with random moves, gives a good estimate for its score.)

Similarly, the AITP systems use a corpus of proven theorems as training data. As the verifier works through a theorem, each step of premise and tactic selection is saved, along with a summary of the state just before the choice is made. One can then use these pairs (state, selection) to train networks to do premise and tactic selection.

The better developed systems (Coq, HOL) have large libraries with 30,000–70,000 (short) theorems. This is enough training data to achieve success rates in proving similar theorems (a held-out testing set) of around 75% (it was 50% two years ago).

A very successful approach to AI game play is reinforcement learning (RL), as famously used by AlphaGo. Its central parts are a pair of neural networks, one to choose moves and the other to evaluate the score of game positions. The original AlphaGo used a corpus of human games for initial training, and then generated games by self-play. (Another important element is Monte Carlo tree search – it turns out for Go that playing out a game many times with random moves, gives a good estimate for its score.)

Similarly, the AITP systems use a corpus of proven theorems as training data. As the verifier works through a theorem, each step of premise and tactic selection is saved, along with a summary of the state just before the choice is made. One can then use these pairs (state, selection) to train networks to do premise and tactic selection.

The better developed systems (Coq, HOL) have large libraries with 30,000–70,000 (short) theorems. This is enough training data to achieve success rates in proving similar theorems (a held-out testing set) of around 75% (it was 50% two years ago).

Auto-formalization

The total corpus of mathematical texts is much larger of course – there are about 1.5 million papers just on arXiv. Building on the success of neural networks for machine translation, could we develop a system which translates "informal" mathematical text to formal mathematics?

These experiments are in very early days, see for example `https://arxiv.org/abs/1611.09703` by Kaliszyk *et al*. One problem is that there is no sizable corpus of aligned informal and formal mathematics to use as training data. So far this is dealt with by "informalizing" a formal corpus.

To my mind, a deeper problem is that mathematical texts are almost never self-contained, and a formalization cannot make much sense without the formal definitions of the concepts it refers to. In the best cases a text will only refer to standard concepts, so having a library as in Formal Abstracts would be a great help. But in many (most?) cases research papers refer to other research papers.

Auto-formalization

The total corpus of mathematical texts is much larger of course – there are about 1.5 million papers just on arXiv. Building on the success of neural networks for machine translation, could we develop a system which translates “informal” mathematical text to formal mathematics?

These experiments are in very early days, see for example `https://arxiv.org/abs/1611.09703` by Kaliszyk *et al.* One problem is that there is no sizable corpus of aligned informal and formal mathematics to use as training data. So far this is dealt with by “informalizing” a formal corpus.

To my mind, a deeper problem is that mathematical texts are almost never self-contained, and a formalization cannot make much sense without the formal definitions of the concepts it refers to. In the best cases a text will only refer to standard concepts, so having a library as in Formal Abstracts would be a great help. But in many (most?) cases research papers refer to other research papers.

Auto-formalization

The total corpus of mathematical texts is much larger of course – there are about 1.5 million papers just on arXiv. Building on the success of neural networks for machine translation, could we develop a system which translates "informal" mathematical text to formal mathematics?

These experiments are in very early days, see for example `https://arxiv.org/abs/1611.09703` by Kaliszyk *et al*. One problem is that there is no sizable corpus of aligned informal and formal mathematics to use as training data. So far this is dealt with by "informalizing" a formal corpus.

To my mind, a deeper problem is that mathematical texts are almost never self-contained, and a formalization cannot make much sense without the formal definitions of the concepts it refers to. In the best cases a text will only refer to standard concepts, so having a library as in Formal Abstracts would be a great help. But in many (most?) cases research papers refer to other research papers.

Reinforcement learning

The big success of AlphaGo came from its ability to generate its own training data by self-play. Amazingly, it turned out that human games were not needed – AlphaGo Zero achieved superhuman skill without them. Could we make a theorem prover do self-play?

At first, one might say that to "win a game" is equivalent to proving a theorem. However this is simplistic as every step of a deduction proves a new logical statement. Somehow the results have to be scored according to how "significant," "interesting" or "useful" the statement is, or how close the new statement is to a significant result.

Only rewarding the theorems considered interesting or significant by humans may not be giving the computer enough feedback. So, it may be necessary to give the computer its own ability to judge what is interesting. From an ML point of view this is just another scoring function which could be learned.

Reinforcement learning

The big success of AlphaGo came from its ability to generate its own training data by self-play. Amazingly, it turned out that human games were not needed – AlphaGo Zero achieved superhuman skill without them. Could we make a theorem prover do self-play?

At first, one might say that to “win a game” is equivalent to proving a theorem. However this is simplistic as every step of a deduction proves a new logical statement. Somehow the results have to be scored according to how “significant,” “interesting” or “useful” the statement is, or how close the new statement is to a significant result.

Only rewarding the theorems considered interesting or significant by humans may not be giving the computer enough feedback. So, it may be necessary to give the computer its own ability to judge what is interesting. From an ML point of view this is just another scoring function which could be learned.

Reinforcement learning

The big success of AlphaGo came from its ability to generate its own training data by self-play. Amazingly, it turned out that human games were not needed – AlphaGo Zero achieved superhuman skill without them. Could we make a theorem prover do self-play?

At first, one might say that to "win a game" is equivalent to proving a theorem. However this is simplistic as every step of a deduction proves a new logical statement. Somehow the results have to be scored according to how "significant," "interesting" or "useful" the statement is, or how close the new statement is to a significant result.

Only rewarding the theorems considered interesting or significant by humans may not be giving the computer enough feedback. So, it may be necessary to give the computer its own ability to judge what is interesting. From an ML point of view this is just another scoring function which could be learned.

What is “interesting” mathematics? In Lenat’s AM system (1976) this was defined by hand-coded heuristics.

Perhaps given a large semi-formal corpus, this could be inferred by similarity of a statement to the statements in the corpus. After all, people try to make interesting statements and avoid uninteresting ones.

Rather than make an *a priori* definition of interesting, one can say that an interesting concept is one which aids reasoning. To the extent that the system can judge the complexity of its proofs, then a new statement which makes many proofs simpler is *ipso facto* interesting.

One could consider efficacy at more general tasks. Perhaps textbook problems would be a good source. As another example, given a mathematical definitions such as “finite group,” can the system take a pair of randomly chosen examples and efficiently prove that they are isomorphic or not isomorphic. As an even harder test, can the system enumerate groups with up to k elements?

What is “interesting” mathematics? In Lenat’s AM system (1976) this was defined by hand-coded heuristics.

Perhaps given a large semi-formal corpus, this could be inferred by similarity of a statement to the statements in the corpus. After all, people try to make interesting statements and avoid uninteresting ones.

Rather than make an *a priori* definition of interesting, one can say that an interesting concept is one which aids reasoning. To the extent that the system can judge the complexity of its proofs, then a new statement which makes many proofs simpler is *ipso facto* interesting.

One could consider efficacy at more general tasks. Perhaps textbook problems would be a good source. As another example, given a mathematical definitions such as “finite group,” can the system take a pair of randomly chosen examples and efficiently prove that they are isomorphic or not isomorphic. As an even harder test, can the system enumerate groups with up to k elements?

What is "interesting" mathematics? In Lenat's AM system (1976) this was defined by hand-coded heuristics.

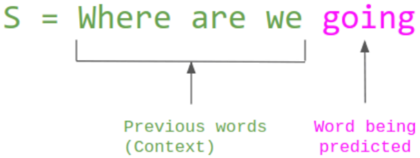
Perhaps given a large semi-formal corpus, this could be inferred by similarity of a statement to the statements in the corpus. After all, people try to make interesting statements and avoid uninteresting ones.

Rather than make an *a priori* definition of interesting, one can say that an interesting concept is one which aids reasoning. To the extent that the system can judge the complexity of its proofs, then a new statement which makes many proofs simpler is *ipso facto* interesting.

One could consider efficacy at more general tasks. Perhaps textbook problems would be a good source. As another example, given a mathematical definitions such as "finite group," can the system take a pair of randomly chosen examples and efficiently prove that they are isomorphic or not isomorphic. As an even harder test, can the system enumerate groups with up to k elements?

Large Language Models

The term language model usually refers to a probabilistic model, for example a model for the conditional probability that a sequence of words is followed by another specified word.

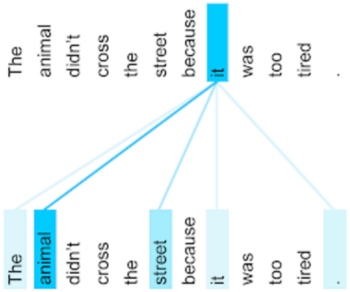


P(S) = P(Where) x P(are | Where) x P(we | Where are) x P(going | Where are we)

Such a probability description can be modeled and fit to a large corpus of text, for example the 100's of gigabytes of natural language on the web.

This type of language model has been studied for a long time, for example in the 1993 work of Brown *et al* on statistical machine translation.

In 2017 a neural network architecture called **attention** was proposed by Vaswani *et al*. The basic idea is to have a neural network for each word, which can take its input from any of the words in the previous layer depending on a learned measure of relevance. This dramatically improved performance.



Large language models such as GPT-3 and PaLM are trained on large fractions of the text available on the internet, 100's of gigabytes. This can also include Github repositories (more 100's of gigabytes).

The resulting models can be used to generate text which looks eerily similar to what a human would write, at least if one doesn't look too closely (it lacks coherence and can have elementary mistakes). They are also the basis of systems such as Google Translate.

The models have amazing capabilities far beyond what anyone expected. Among the impressive recent developments are Codex and AlphaCode, systems which translate natural language descriptions of a problem to runnable code in Python or other languages.

Language models can even solve math and physics word problems at an advanced high school or early undergraduate level. Google's Minerva system (arXiv:2206.14858) is a language model fine-tuned on text from arXiv and other mathematical sources, and gets almost 50% accuracy.

```
def is_prime(n):
    """Return true if a given number is prime, and
        false otherwise.
    >>> is_prime(6)
    False
    >>> is_prime(101)
    True
    >>> is_prime(11)
    True
    >>> is_prime(13441)
    True
    >>> is_prime(61)
    True
    >>> is_prime(4)
    False
    >>> is_prime(1)
    False
    """
```

```
### COMPLETION 6 (CORRECT): ###
prime = True
if n == 1:
    return False
for i in range(2, n):
    if n % i == 0:
        prime = False
return prime
```

```
### COMPLETION 7 (WRONG): ###
for i in range(2, n):
    if n % i == 0:
        return False
return True
```

Codex - an example from Chen et al 2021 (arXiv:2107.03374)

Natural language	Suppose a, b, c are the sides of a triangle. Prove that $$a^2(b+c-a)+b^2(c+a-b)+c^2(a+b-c) \leq 3abc$$

Model proof	`theorem imo_1964_p2`
	`  (a b c : ℝ)`
	`  (h₀ : 0 < a ∧ 0 < b ∧ 0 < c)`
	`  (h₁ : c < a + b)`
	`  (h₂ : b < a + c)`
	`  (h₃ : a < b + c) :`
	`  a^2 * (b + c - a) + b^2 * (c + a - b) + c^2 * (a + b - c)`
	`    ≤ 3 * a * b * c :=`
	`begin`
	`  nlinarith [sq_nonneg (b - a), sq_nonneg (c - b),`
	`    sq_nonneg (a - c), sq_nonneg (c - a)]`
	`end`

Comments	The model is able to close an IMO problem in one-line. It correctly provides exogenous arguments to `nlinarith`, which are necessary to close the goal. Note that either one of the last two arguments in the sequence `[sq_nonneg (b - a), sq_nonneg (c - b), sq_nonneg (a - c), sq_nonneg (c - a)]` can be omitted.

This is an example of the language model approach (Polu et al, 2022). Their GPT-f system can accept natural language mathematics problems (here from the IMO) and, at least sometimes, find the solutions with formal proofs.

Question: A line parallel to $y = 4x + 6$ passes through $(5, 10)$. What is the y-coordinate of the point where this line crosses the y-axis?

Human solution: Since the line is parallel to $y = 4x + 6$, we know that its slope is 4. Thus the equation of the line is $y = 4x + b$ where b is the y-intercept. Plugging $(5, 10)$ into this gives $10 = 4 \cdot 5 + b \Rightarrow b = \boxed{-10}$ which is what we wanted.

Minerva output: The line is parallel to $y = 4x + 6$, which means that it has the same slope as $y = 4x + 6$. Since the line passes through $(5, 10)$, we can use the point-slope form of a line to find the equation of the line:

$$\begin{aligned} y - 10 &= 4(x - 5) \\ y - 10 &= 4x - 20 \\ y &= 4x - 10 \end{aligned}$$

Now that we have the equation of the line, we can find the y-coordinate of the point where the line crosses the y-axis by substituting $x = 0$ into the equation:

$$y = 4 \cdot 0 - 10 = \boxed{-10}.$$

Figure: From Lewkowycz *et al*, arXiv:2206.14858.

Summary and conclusions

Some major lines of progress in computers and mathematics:

- General advances in hardware and software
- More and larger knowledge repositories, with better metadata
- Neural networks and deep learning
- Interactive theorem verification, theorem libraries and formal abstracts
- New styles of collaboration along the lines of the Stack project, Polymath, ...

Speculations and opportunities

In the end I don't think there are very clear predictions to make for 2030, but let me go out on a limb and make some anyways.

- There will be textbooks which integrate computation – not just numerics but many of the new technologies we discuss – in groundbreaking ways, making today's textbooks look dated.
- Significant new discoveries and proofs will continue to be made using computational experiment and numerical methods, but of the general character we have seen: solving intricate problems by combinatorial search, producing large datasets leading to statistical conjectures, *etc.*.
- ITVs will be much easier to use, thanks to AI automation. An introductory course in theorem proving will be a common advanced undergraduate offering. They will have a status much like computer algebra systems now – a convenient tool that some people rely on and many people use (say) once a month.

Speculations and opportunities

In the end I don't think there are very clear predictions to make for 2030, but let me go out on a limb and make some anyways.

- There will be textbooks which integrate computation – not just numerics but many of the new technologies we discuss – in groundbreaking ways, making today's textbooks look dated.
- Significant new discoveries and proofs will continue to be made using computational experiment and numerical methods, but of the general character we have seen: solving intricate problems by combinatorial search, producing large datasets leading to statistical conjectures, *etc.*.
- ITVs will be much easier to use, thanks to AI automation. An introductory course in theorem proving will be a common advanced undergraduate offering. They will have a status much like computer algebra systems now – a convenient tool that some people rely on and many people use (say) once a month.

Speculations and opportunities

In the end I don't think there are very clear predictions to make for 2030, but let me go out on a limb and make some anyways.

- There will be textbooks which integrate computation – not just numerics but many of the new technologies we discuss – in groundbreaking ways, making today's textbooks look dated.
- Significant new discoveries and proofs will continue to be made using computational experiment and numerical methods, but of the general character we have seen: solving intricate problems by combinatorial search, producing large datasets leading to statistical conjectures, *etc.*.
- ITVs will be much easier to use, thanks to AI automation. An introductory course in theorem proving will be a common advanced undergraduate offering. They will have a status much like computer algebra systems now – a convenient tool that some people rely on and many people use (say) once a month.

Speculations and opportunities

In the end I don't think there are very clear predictions to make for 2030, but let me go out on a limb and make some anyways.

- There will be textbooks which integrate computation – not just numerics but many of the new technologies we discuss – in groundbreaking ways, making today's textbooks look dated.
- Significant new discoveries and proofs will continue to be made using computational experiment and numerical methods, but of the general character we have seen: solving intricate problems by combinatorial search, producing large datasets leading to statistical conjectures, *etc.*.
- ITVs will be much easier to use, thanks to AI automation. An introductory course in theorem proving will be a common advanced undergraduate offering. They will have a status much like computer algebra systems now – a convenient tool that some people rely on and many people use (say) once a month.

- A large database of formal mathematics will exist, built by some combination of human and automated work. Perhaps the overall organization will be created or tuned by humans, while the bulk of the formalization will be automatic.
- Semantic mathematical search will be a standard part of our literature searches. At least one nontrivial connection between different mathematical fields will be discovered this way.
- At present it is hard to reuse code from math/physics projects – this problem will be largely solved.

But in 2030 will computers have invented or proven any major result by themselves? Math AGI, either based on the human mathematical literature, or which trains itself *ab initio*, is an old dream.

- A large database of formal mathematics will exist, built by some combination of human and automated work. Perhaps the overall organization will be created or tuned by humans, while the bulk of the formalization will be automatic.
- Semantic mathematical search will be a standard part of our literature searches. At least one nontrivial connection between different mathematical fields will be discovered this way.
- At present it is hard to reuse code from math/physics projects – this problem will be largely solved.

But in 2030 will computers have invented or proven any major result by themselves? Math AGI, either based on the human mathematical literature, or which trains itself *ab initio*, is an old dream.

- A large database of formal mathematics will exist, built by some combination of human and automated work. Perhaps the overall organization will be created or tuned by humans, while the bulk of the formalization will be automatic.
- Semantic mathematical search will be a standard part of our literature searches. At least one nontrivial connection between different mathematical fields will be discovered this way.
- At present it is hard to reuse code from math/physics projects – this problem will be largely solved.

But in 2030 will computers have invented or proven any major result by themselves? Math AGI, either based on the human mathematical literature, or which trains itself *ab initio*, is an old dream.

- A large database of formal mathematics will exist, built by some combination of human and automated work. Perhaps the overall organization will be created or tuned by humans, while the bulk of the formalization will be automatic.
- Semantic mathematical search will be a standard part of our literature searches. At least one nontrivial connection between different mathematical fields will be discovered this way.
- At present it is hard to reuse code from math/physics projects – this problem will be largely solved.

But in 2030 will computers have invented or proven any major result by themselves? Math AGI, either based on the human mathematical literature, or which trains itself *ab initio*, is an old dream.

While progress in AI is rapid, we don't have a good way to guess how far away this dream might be. But here are some thoughts:

- Math reasoning – at the level of individual steps – is not likely to be easier than general reasoning in other large domains.
- Other domains may have advantages, such as more training data.
- The advantages of math as a domain are that one can use arbitrarily long chains of reasoning, and that success depends much less on abilities other than logical reasoning. So, even if general reasoning capability is developed in many domains at once, it will have major consequences in math before most fields.
- Significant advances are being made in general reasoning, and breakthroughs may happen soon.
- The full consequences of a breakthrough will take around ten years to realize. For example, MCTS was introduced in computer Go around 2006, and produced significant improvements by 2009, leading to AlphaGo in 2016.