Automating Mathematics?

Siddhartha Gadgil

Department of Mathematics,
Indian Institute of Science.

July 31, 2018

Goal

- To equip computers with the ability to perform all major tasks involved in the discovery and proof of mathematical results and concepts by mathematicians and the mathematics community, at a level at least comparable to humans.

Goal

- To equip computers with the ability to perform all major tasks involved in the discovery and proof of mathematical results and concepts by mathematicians and the mathematics community, at a level at least comparable to humans.
- We must have, if necessary invent, objective measures to see whether, and how well the tasks are performed.

Goal

- To equip computers with the ability to perform all major tasks involved in the discovery and proof of mathematical results and concepts by mathematicians and the mathematics community, at a level at least comparable to humans.
- We must have, if necessary invent, objective measures to see whether, and how well the tasks are performed.
- It may be useful to invent tasks as exercises.

Outline

1. Computer Assisted Mathematics

Outline

1. Computer Assisted Mathematics

2. Mathematical Tasks

Outline

1. Computer Assisted Mathematics
2. Mathematical Tasks
3. Artificial Inteligence elsewhere

Outline

1. Computer Assisted Mathematics
2. Mathematical Tasks
3. Artificial Inteligence elsewhere
4. Mathematical Tasks revisited

Outline

1. Computer Assisted Mathematics

2. Mathematical Tasks

3. Artificial Inteligence elsewhere

4. Mathematical Tasks revisited

5. Conclusions

Computer Assisted Mathematics

What computers can do

- Numerical computation.

What computers can do

- Numerical computation.
- Enumeration.

What computers can do

- Numerical computation.
- Enumeration.
- Symbolic algebra; computational algebra.

What computers can do

- Numerical computation.
- Enumeration.
- Symbolic algebra; computational algebra.
- Exact real number arithmetic.

What computers can do

- Numerical computation.
- Enumeration.
- Symbolic algebra; computational algebra.
- Exact real number arithmetic.
- Linear programming.

What computers can do

- Numerical computation.
- Enumeration.
- Symbolic algebra; computational algebra.
- Exact real number arithmetic.
- Linear programming.
- SAT solvers.

What computers can do

- Numerical computation.
- Enumeration.
- Symbolic algebra; computational algebra.
- Exact real number arithmetic.
- Linear programming.
- SAT solvers.
- Compact Enumeration.

Some computer-assisted proofs

- Four colour theorem.

Some computer-assisted proofs

- Four colour theorem.
- Kepler conjecture.

Some computer-assisted proofs

- Four colour theorem.
- Kepler conjecture.
- Boolean Pythagorean triples problem.

Some computer-assisted proofs

- Four colour theorem.
- Kepler conjecture.
- Boolean Pythagorean triples problem.
- Existence of Lorenz attractor.

Some computer-assisted proofs

- Four colour theorem.
- Kepler conjecture.
- Boolean Pythagorean triples problem.
- Existence of Lorenz attractor.
- The 290 Theorem for integral quadratic forms.

Robbins Conjecture: Deductive proofs

- Robbins conjecture was a conjectural characterization of Boolean algebras in terms of associativity and commutativity of $\vee$ and the Robbins equation
$$\neg(\neg(a \vee b) \vee \neg(a \vee \neg b)) = a.$$

Robbins Conjecture: Deductive proofs

- Robbins conjecture was a conjectural characterization of Boolean algebras in terms of associativity and commutativity of $\vee$ and the Robbins equation

$$\neg(\neg(a \vee b) \vee \neg(a \vee \neg b)) = a.$$

- This was conjectured in the 1930s.

Robbins Conjecture: Deductive proofs

- Robbins conjecture was a conjectural characterization of Boolean algebras in terms of associativity and commutativity of $\vee$ and the Robbins equation
$$\neg(\neg(a \vee b) \vee \neg(a \vee \neg b)) = a.$$
- This was conjectured in the 1930s.
- It was finally proved in 1996 using the automated theorem prover EQP.

Robbins Conjecture: Deductive proofs

- Robbins conjecture was a conjectural characterization of Boolean algebras in terms of associativity and commutativity of $\vee$ and the Robbins equation
$$\neg(\neg(a \vee b) \vee \neg(a \vee \neg b)) = a.$$
- This was conjectured in the 1930s.
- It was finally proved in 1996 using the automated theorem prover EQP.
- This is a Resolution Theorem Prover with Paramodulation.

Interactive Theorem Proving

- Interactive theorem provers such as Coq, Isabelle and Lean fill in details of and verify results.

Interactive Theorem Proving

- Interactive theorem provers such as Coq, Isabelle and Lean fill in details of and verify results.
- In practice these have been used (so far) in formalizing proofs, not discovery.

Interactive Theorem Proving

- Interactive theorem provers such as Coq, Isabelle and Lean fill in details of and verify results.
- In practice these have been used (so far) in formalizing proofs, not discovery.
- The greatest success so far has been the formal proof of the Feit-Thompson theorem by Georges Gonthier.

Homogeneous length functions on groups

Homogeneous length functions on groups

Question (Apoorva Khare via Terence Tao)

Is there a function $l : \langle \alpha, \beta \rangle \to [0, \infty)$ *on the free group on two generators such that*

Homogeneous length functions on groups

Question (Apoorva Khare via Terence Tao)

Is there a function $l : \langle \alpha, \beta \rangle \to [0, \infty)$ on the free group on two generators such that

- $l(g) = 0$ *if and only if* $g = e$ *(positivity).*

Homogeneous length functions on groups

Question (Apoorva Khare via Terence Tao)

Is there a function $l : \langle \alpha, \beta \rangle \to [0, \infty)$ on the free group on two generators such that

- $l(g) = 0$ *if and only if* $g = e$ *(positivity).*
- $l(g^{-1}) = l(g)$ *for all* $g \in \langle \alpha, \beta \rangle$.

Homogeneous length functions on groups

Question (Apoorva Khare via Terence Tao)

Is there a function $l : \langle \alpha, \beta \rangle \to [0, \infty)$ on the free group on two generators such that

- $l(g) = 0$ *if and only if* $g = e$ *(positivity).*
- $l(g^{-1}) = l(g)$ *for all* $g \in \langle \alpha, \beta \rangle$*.*
- $l(gh) \leq l(g) + l(h)$ *for all* $g, h \in \langle \alpha, \beta \rangle$*.*

Homogeneous length functions on groups

Question (Apoorva Khare via Terence Tao)

Is there a function $l : \langle \alpha, \beta \rangle \to [0, \infty)$ on the free group on two generators such that

- $l(g) = 0$ *if and only if* $g = e$ *(positivity).*
- $l(g^{-1}) = l(g)$ *for all* $g \in \langle \alpha, \beta \rangle$.
- $l(gh) \leq l(g) + l(h)$ *for all* $g, h \in \langle \alpha, \beta \rangle$.
- $l(ghg^{-1}) = l(h)$ *for all* $g, h \in \langle \alpha, \beta \rangle$.

Homogeneous length functions on groups

Question (Apoorva Khare via Terence Tao)

Is there a function $l : \langle \alpha, \beta \rangle \to [0, \infty)$ on the free group on two generators such that

- $l(g) = 0$ *if and only if* $g = e$ *(positivity).*
- $l(g^{-1}) = l(g)$ *for all* $g \in \langle \alpha, \beta \rangle$.
- $l(gh) \leq l(g) + l(h)$ *for all* $g, h \in \langle \alpha, \beta \rangle$.
- $l(ghg^{-1}) = l(h)$ *for all* $g, h \in \langle \alpha, \beta \rangle$.
- $l(g^n) = nl(g)$ *for all* $g \in \langle \alpha, \beta \rangle$, $n \in \mathbb{Z}$.

The Quest

- On Saturday, December 16, 2017, Terence Tao posted this question on his blog for crowdsourcing.

The Quest

- On Saturday, December 16, 2017, Terence Tao posted this question on his blog for crowdsourcing.
- Over the next 4-5 days, by work of many people,

The Quest

- On Saturday, December 16, 2017, Terence Tao posted this question on his blog for crowdsourcing.
- Over the next 4-5 days, by work of many people,
  - there were many (failed, but instructive) attempts to construct such length functions,

The Quest

- On Saturday, December 16, 2017, Terence Tao posted this question on his blog for crowdsourcing.
- Over the next 4-5 days, by work of many people,
  - there were many (failed, but instructive) attempts to construct such length functions,
  - leading to the general feeling that $l([\alpha, \beta]) = 0$;

The Quest

- On Saturday, December 16, 2017, Terence Tao posted this question on his blog for crowdsourcing.
- Over the next 4-5 days, by work of many people,
  - there were many (failed, but instructive) attempts to construct such length functions,
  - leading to the general feeling that $l([\alpha, \beta]) = 0$;
  - increasingly sharp bounds and methods of combining bounds, but no visible path to $l([\alpha, \beta]) = 0$.

The Quest

- On Saturday, December 16, 2017, Terence Tao posted this question on his blog for crowdsourcing.
- Over the next 4-5 days, by work of many people,
  - there were many (failed, but instructive) attempts to construct such length functions,
  - leading to the general feeling that $l([\alpha, \beta]) = 0$;
  - increasingly sharp bounds and methods of combining bounds, but no visible path to $l([\alpha, \beta]) = 0$.
- On Thursday morning I posted a proof of a computer-assisted bound.

Proof which I posted online

Proof which I posted online

Proof of a bound on $l([\alpha, \beta])$ for l a homogeneous, conjugacy invariant length function with $l(\alpha), l(\beta) \leq 1$.

Proof which I posted online

Proof of a bound on $l([\alpha, \beta])$ for l a homogeneous, conjugacy invariant length function with $l(\alpha), l(\beta) \leq 1$.

- $|\bar{a}| \leq 1.0$
- $|\bar{b}\bar{a}b| \leq 1.0$ using $|\bar{a}| \leq 1.0$
- $|\bar{b}| \leq 1.0$
- $|a\bar{b}\bar{a}| \leq 1.0$ using $|\bar{b}| \leq 1.0$
- $|\bar{a}\bar{b}ab\bar{a}\bar{b}| \leq 2.0$ using $|\bar{a}\bar{b}a| \leq 1.0$ and $|b\bar{a}\bar{b}| \leq 1.0$
- ... (119 lines)
- $|ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}| \leq 13.859649122807017$ using $|ab\bar{a}| \leq 1.0$ and $|\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}| \leq 12.859649122807017$
- $|ab\bar{a}\bar{b}| \leq 0.8152734778121775$ using $|ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}| \leq 13.859649122807017$ by taking 17th power.

Proof which I posted online

Proof of a bound on $l([\alpha, \beta])$ for l a homogeneous, conjugacy invariant length function with $l(\alpha), l(\beta) \leq 1$.

- $|\bar{a}| \leq 1.0$
- $|\bar{b}\bar{a}b| \leq 1.0$ using $|\bar{a}| \leq 1.0$
- $|\bar{b}| \leq 1.0$
- $|a\bar{b}\bar{a}| \leq 1.0$ using $|\bar{b}| \leq 1.0$
- $|\bar{a}\bar{b}ab\bar{a}\bar{b}| \leq 2.0$ using $|\bar{a}\bar{b}a| \leq 1.0$ and $|b\bar{a}\bar{b}| \leq 1.0$
- ... (119 lines)
- $|ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}| \leq$ 13.859649122807017 using $|ab\bar{a}| \leq 1.0$ and $|\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}| \leq$ 12.859649122807017
- $|ab\bar{a}\bar{b}| \leq 0.8152734778121775$ using $|ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}ab\bar{a}\bar{b}| \leq$ 13.859649122807017 by taking 17th power.

i.e., $l(\alpha, \beta) \leq 0.8152734778121775$

- The computer-generated proof was studied by Pace Nielsen, who extracted the internal repetition trick.

- The computer-generated proof was studied by Pace Nielsen, who extracted the internal repetition trick.
- This was extended by Pace Nielsen and Tobias Fritz and generalized by Terence Tao.

- The computer-generated proof was studied by Pace Nielsen, who extracted the internal repetition trick.
- This was extended by Pace Nielsen and Tobias Fritz and generalized by Terence Tao.
- From this Fritz obtained the key lemma:

- The computer-generated proof was studied by Pace Nielsen, who extracted the internal repetition trick.
- This was extended by Pace Nielsen and Tobias Fritz and generalized by Terence Tao.
- From this Fritz obtained the key lemma:

Lemma

Let $f(m, k) = l(x^m[x, y]^k)$. *Then*

$$f(m, k) \leq \frac{f(m-1, k) + f(m+1, k-1)}{2}.$$

- The computer-generated proof was studied by Pace Nielsen, who extracted the internal repetition trick.
- This was extended by Pace Nielsen and Tobias Fritz and generalized by Terence Tao.
- From this Fritz obtained the key lemma:

Lemma

Let $f(m,k) = l(x^m[x,y]^k)$. *Then*

$$f(m,k) \leq \frac{f(m-1,k) + f(m+1,k-1)}{2}.$$

- A probabilistic argument of Tao showed $l([x,y]) = 0$.

Mathematical Tasks

- Introduce/construct (invent, discover):

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, conjectures, goals, techniques, heuristics.

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, conjectures, goals, techniques, heuristics.
  - By recalling and using existing objects, including by analogy, generalization, instantiation (specialization).

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, conjectures, goals, techniques, heuristics.
  - By recalling and using existing objects, including by analogy, generalization, instantiation (specialization).
  - Experiment and judge plausibility.

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, conjectures, goals, techniques, heuristics.
  - By recalling and using existing objects, including by analogy, generalization, instantiation (specialization).
  - Experiment and judge plausibility.
- Evaluate (judge) based on:

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, conjectures, goals, techniques, heuristics.
  - By recalling and using existing objects, including by analogy, generalization, instantiation (specialization).
  - Experiment and judge plausibility.
- Evaluate (judge) based on:
  - Outcomes – known questions, simple statements with hard proofs, novelty, depth, applications etc.

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, conjectures, goals, techniques, heuristics.
  - By recalling and using existing objects, including by analogy, generalization, instantiation (specialization).
  - Experiment and judge plausibility.
- Evaluate (judge) based on:
  - Outcomes – known questions, simple statements with hard proofs, novelty, depth, applications etc.
  - Derived value – expected to be useful for outcomes.

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, conjectures, goals, techniques, heuristics.
  - By recalling and using existing objects, including by analogy, generalization, instantiation (specialization).
  - Experiment and judge plausibility.
- Evaluate (judge) based on:
  - Outcomes – known questions, simple statements with hard proofs, novelty, depth, applications etc.
  - Derived value – expected to be useful for outcomes.
- (Usually) depending on contexts and goals.

- Digest/Refine: both newly discovered mathematics and known mathematics.

- Digest/Refine: both newly discovered mathematics and known mathematics.
- Develop strategies for proving/solving: intermediate goals, allocate resources.

- Digest/Refine: both newly discovered mathematics and known mathematics.
- Develop strategies for proving/solving: intermediate goals, allocate resources.
- Keep improving.

- Digest/Refine: both newly discovered mathematics and known mathematics.
- Develop strategies for proving/solving: intermediate goals, allocate resources.
- Keep improving.
- Read and digest the literature (some of which is formalized).

- Digest/Refine: both newly discovered mathematics and known mathematics.
- Develop strategies for proving/solving: intermediate goals, allocate resources.
- Keep improving.
- Read and digest the literature (some of which is formalized).
- Handle mathematics *in the large*.

- Digest/Refine: both newly discovered mathematics and known mathematics.
- Develop strategies for proving/solving: intermediate goals, allocate resources.
- Keep improving.
- Read and digest the literature (some of which is formalized).
- Handle mathematics *in the large*.
- Find good and useful proofs, in particular proofs from which we can learn.

Artificial Inteligence elsewhere

Chess and friends

- Playing Chess, etc can be based on

Chess and friends

- Playing Chess, etc can be based on
  - Evaluation of a fixed players position (say White).

Chess and friends

- Playing Chess, etc can be based on
  - Evaluation of a fixed players position (say White).
  - Policy: which sequences of moves to consider.

Chess and friends

- Playing Chess, etc can be based on
  - Evaluation of a fixed players position (say White).
  - Policy: which sequences of moves to consider.
- We evaluate the state at the end of sequences of moves we consider.

Chess and friends

- Playing Chess, etc can be based on
  - Evaluation of a fixed players position (say White).
  - Policy: which sequences of moves to consider.
- We evaluate the state at the end of sequences of moves we consider.
- Using this, we recursively decide the best moves based on alternately maximizing and minimizing.

Programming a Computer for Playing Chess

Programming a Computer for Playing Chess

- Shannon distinguished two kinds of strategies – type A where all moves are considered up to a fixed depth and type B where a refined policy is used.

Programming a Computer for Playing Chess

- Shannon distinguished two kinds of strategies – type A where all moves are considered up to a fixed depth and type B where a refined policy is used.
- Various heuristics, such as quiescence search and $\alpha - \beta$ pruning are used to refine type A engines.

Programming a Computer for Playing Chess

- Shannon distinguished two kinds of strategies – type A where all moves are considered up to a fixed depth and type B where a refined policy is used.
- Various heuristics, such as quiescence search and $\alpha - \beta$ pruning are used to refine type A engines.
- Openings and end-games are instead based on databases.

Programming a Computer for Playing Chess

- Shannon distinguished two kinds of strategies – type A where all moves are considered up to a fixed depth and type B where a refined policy is used.
- Various heuristics, such as quiescence search and $\alpha - \beta$ pruning are used to refine type A engines.
- Openings and end-games are instead based on databases.
- Deep blue (which defeated Kasparov in 1997) and other top chess engines are such systems.

Limitations of Chess Engines

- The policy is very weak, considering almost all moves or only a few.

Limitations of Chess Engines

- The policy is very weak, considering almost all moves or only a few.
- Evaluation is also sub-human, especially when it comes to complex positional values.

Limitations of Chess Engines

- The policy is very weak, considering almost all moves or only a few.
- Evaluation is also sub-human, especially when it comes to complex positional values.
- Chess engines also do not think strategically, i.e., having sub-goals and allocating resources.

Limitations of Chess Engines

- The policy is very weak, considering almost all moves or only a few.
- Evaluation is also sub-human, especially when it comes to complex positional values.
- Chess engines also do not think strategically, i.e., having sub-goals and allocating resources.
- In a different domain, these weaknesses may matter much more than in Chess.

The Game of Go

- In the chinese game of Go,

The Game of Go

- In the chinese game of Go,
    - the number of possible moves is much larger.

The Game of Go

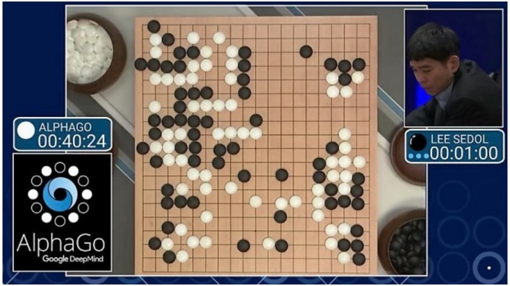
- In the chinese game of Go,
  - the number of possible moves is much larger.
  - It is very difficult to have a good evaluation function.

The Game of Go

- In the chinese game of Go,
  - the number of possible moves is much larger.
  - It is very difficult to have a good evaluation function.
- The Go champion AlphaGo is not an Expert system, but is based instead on Machine Learning.

The Game of Go

- In the chinese game of Go,
  - the number of possible moves is much larger.
  - It is very difficult to have a good evaluation function.
- The Go champion AlphaGo is not an Expert system, but is based instead on Machine Learning.



March 2016 vs Lee Sedol
May 2017 vs Ke Jie

Neural Networks

- A feedforward neural network is a class of functions $f : \mathbb{R}^n \to \mathbb{R}^m$ determined by finitely many real parameters.

Neural Networks

- A feedforward neural network is a class of functions $f : \mathbb{R}^n \to \mathbb{R}^m$ determined by finitely many real parameters.
- Functions in the class are given by compositions of so called layers, which are functions of a specific form.

Neural Networks

- A feedforward neural network is a class of functions $f : \mathbb{R}^n \to \mathbb{R}^m$ determined by finitely many real parameters.
- Functions in the class are given by compositions of so called layers, which are functions of a specific form.
- Each layer is typically the composition of a linear transformation with a sigmoid, e.g., $S(x) = \frac{e^x}{e^x+1}$.

Neural Networks

- A feedforward neural network is a class of functions $f : \mathbb{R}^n \to \mathbb{R}^m$ determined by finitely many real parameters.
- Functions in the class are given by compositions of so called layers, which are functions of a specific form.
- Each layer is typically the composition of a linear transformation with a sigmoid, e.g., $S(x) = \frac{e^x}{e^x+1}$.
- We can optimize functions within this class using a gradient flow layer-by-layer.

AlphaGo

- The policy and value functions of AlphaGo are deep neural networks that were trained.

AlphaGo

- The policy and value functions of AlphaGo are deep neural networks that were trained.
- The policy network was trained by learning to predict the next move from games of expert players.

AlphaGo

- The policy and value functions of AlphaGo are deep neural networks that were trained.
- The policy network was trained by learning to predict the next move from games of expert players.
- The value network was trained by AlphaGo playing against versions of itself.

AlphaGo

- The policy and value functions of AlphaGo are deep neural networks that were trained.
- The policy network was trained by learning to predict the next move from games of expert players.
- The value network was trained by AlphaGo playing against versions of itself.
- AlphaGo considered fewer sequences of moves than Deep Blue.

AlphaGo

- The policy and value functions of AlphaGo are deep neural networks that were trained.
- The policy network was trained by learning to predict the next move from games of expert players.
- The value network was trained by AlphaGo playing against versions of itself.
- AlphaGo considered fewer sequences of moves than Deep Blue.
- AlphaGo came up with unexpected moves.

Word2Vec : Representation learning

- Rather than just treating words as equal or unequal, we associate vectors to them to capture semantics.

Word2Vec : Representation learning

- Rather than just treating words as equal or unequal, we associate vectors to them to capture semantics.
- To do this, we set up the task of predicting the 4 immediate neighbours of a word.

Word2Vec : Representation learning

- Rather than just treating words as equal or unequal, we associate vectors to them to capture semantics.
- To do this, we set up the task of predicting the 4 immediate neighbours of a word.
- We optimize solutions that are compositions of

Word2Vec : Representation learning

- Rather than just treating words as equal or unequal, we associate vectors to them to capture semantics.
- To do this, we set up the task of predicting the 4 immediate neighbours of a word.
- We optimize solutions that are compositions of
  - embeddings of words in $\mathbb{R}^n$ (representations).

Word2Vec : Representation learning

- Rather than just treating words as equal or unequal, we associate vectors to them to capture semantics.
- To do this, we set up the task of predicting the 4 immediate neighbours of a word.
- We optimize solutions that are compositions of
  - embeddings of words in $\mathbb{R}^n$ (representations).
  - functions on $\mathbb{R}^n$.

Word2Vec : Representation learning

- Rather than just treating words as equal or unequal, we associate vectors to them to capture semantics.
- To do this, we set up the task of predicting the 4 immediate neighbours of a word.
- We optimize solutions that are compositions of
  - embeddings of words in $\mathbb{R}^n$ (representations).
  - functions on $\mathbb{R}^n$.
- The vectors capture analogy relations:

$$king - man + woman \approx queen.$$

AlphaGo Zero

- In October 2017, Google DeepMind (the makers of AlphaGo), introduced AlphaGo Zero, a Go playing program much stronger than AlphaGo.

AlphaGo Zero

- In October 2017, Google DeepMind (the makers of AlphaGo), introduced AlphaGo Zero, a Go playing program much stronger than AlphaGo.
- This learnt purely by self-play with zero data.

AlphaGo Zero

- In October 2017, Google DeepMind (the makers of AlphaGo), introduced AlphaGo Zero, a Go playing program much stronger than AlphaGo.
- This learnt purely by self-play with zero data.
- The policy and value networks used a common representation of the Go board.

AlphaGo Zero

- In October 2017, Google DeepMind (the makers of AlphaGo), introduced AlphaGo Zero, a Go playing program much stronger than AlphaGo.
- This learnt purely by self-play with zero data.
- The policy and value networks used a common representation of the Go board.
- In December 2017, this was generalized to AlphaZero, which defeated a top Chess program.

AlphaGo Zero

- In October 2017, Google DeepMind (the makers of AlphaGo), introduced AlphaGo Zero, a Go playing program much stronger than AlphaGo.
- This learnt purely by self-play with zero data.
- The policy and value networks used a common representation of the Go board.
- In December 2017, this was generalized to AlphaZero, which defeated a top Chess program.
- AlphaZero played a bold positional game.

Zero Shot translation

- Instead of training a separate translator between every pair of languages, Google switched to a common network with input labelled by language.

Zero Shot translation

- Instead of training a separate translator between every pair of languages, Google switched to a common network with input labelled by language.
- This could translate between pairs of languages with no training for that pair.

Zero Shot translation

- Instead of training a separate translator between every pair of languages, Google switched to a common network with input labelled by language.
- This could translate between pairs of languages with no training for that pair.
- The system has an internal representation which seems to be based on meanings of sentences.

Generative Adversarial Networks

- These consist of a pair of networks, contesting with each other in a zero-sum game framework.

Generative Adversarial Networks

- These consist of a pair of networks, contesting with each other in a zero-sum game framework.
- One network generates candidates (generative) and the other evaluates them (discriminative).

Generative Adversarial Networks

- These consist of a pair of networks, contesting with each other in a zero-sum game framework.
- One network generates candidates (generative) and the other evaluates them (discriminative).
- The generative network's training objective is to increase the error rate of the discriminative network

Generative Adversarial Networks

- These consist of a pair of networks, contesting with each other in a zero-sum game framework.
- One network generates candidates (generative) and the other evaluates them (discriminative).
- The generative network's training objective is to increase the error rate of the discriminative network
- For example the discriminative network tries to distinguish between real images and synthetic ones generated by the generative network.

Generative Query Network

- In a simulated 3D environment with random light sources, observed 2D images from a few positions.

Generative Query Network

- In a simulated 3D environment with random light sources, observed 2D images from a few positions.
- Had to show the image from a new position.

Generative Query Network

- In a simulated 3D environment with random light sources, observed 2D images from a few positions.
- Had to show the image from a new position.
- The GQN model composed of two parts: a representation network and a generation network.

Generative Query Network

- In a simulated 3D environment with random light sources, observed 2D images from a few positions.
- Had to show the image from a new position.
- The GQN model composed of two parts: a representation network and a generation network.
- The representation network captures important elements, such as object positions, colours and the room layout, in a concise distributed representation.

Generative Query Network

- In a simulated 3D environment with random light sources, observed 2D images from a few positions.
- Had to show the image from a new position.
- The GQN model composed of two parts: a representation network and a generation network.
- The representation network captures important elements, such as object positions, colours and the room layout, in a concise distributed representation.
- The representations showed compositional behaviour.

What Artificial Inteligence can do

- Generate solutions that we can see are good.

What Artificial Inteligence can do

- Generate solutions that we can see are good.
- Judge value based on future likely outcomes.

What Artificial Inteligence can do

- Generate solutions that we can see are good.
- Judge value based on future likely outcomes.
- Show originality.

What Artificial Inteligence can do

- Generate solutions that we can see are good.
- Judge value based on future likely outcomes.
- Show originality.
- Learn things for which we depend on tacit knowledge: "we know more than we can say."

What Artificial Inteligence can do

- Generate solutions that we can see are good.
- Judge value based on future likely outcomes.
- Show originality.
- Learn things for which we depend on tacit knowledge: "we know more than we can say."
- Work with limited and/or unstructured data.

What Artificial Inteligence can do

- Generate solutions that we can see are good.
- Judge value based on future likely outcomes.
- Show originality.
- Learn things for which we depend on tacit knowledge: “we know more than we can say.”
- Work with limited and/or unstructured data.
- Organize observations naturally and efficiently.

Mathematical Tasks revisited

Learning to do Mathematics

- We must learn

Learning to do Mathematics

- We must learn
  - Mathematics, i.e., a body of knowledge.

Learning to do Mathematics

- We must learn
  - Mathematics, i.e., a body of knowledge.
  - How to prove theorems etc using this knowledge.

Learning to do Mathematics

- We must learn
  - Mathematics, i.e., a body of knowledge.
  - How to prove theorems etc using this knowledge.
  - How to assimilate results to extend our knowledge.

Learning to do Mathematics

- We must learn
  - Mathematics, i.e., a body of knowledge.
  - How to prove theorems etc using this knowledge.
  - How to assimilate results to extend our knowledge.
- The body of knowledge should be

Learning to do Mathematics

- We must learn
  - Mathematics, i.e., a body of knowledge.
  - How to prove theorems etc using this knowledge.
  - How to assimilate results to extend our knowledge.
- The body of knowledge should be
  - Efficient at proving theorems (relative entropy).

Learning to do Mathematics

- We must learn
  - Mathematics, i.e., a body of knowledge.
  - How to prove theorems etc using this knowledge.
  - How to assimilate results to extend our knowledge.
- The body of knowledge should be
  - Efficient at proving theorems (relative entropy).
  - Parsimonious (entropy).

Learning to do Mathematics

- We must learn
  - Mathematics, i.e., a body of knowledge.
  - How to prove theorems etc using this knowledge.
  - How to assimilate results to extend our knowledge.
- The body of knowledge should be
  - Efficient at proving theorems (relative entropy).
  - Parsimonious (entropy).
  - Structured (foundations, representation learning).

Learning to do Mathematics

- We must learn
  - Mathematics, i.e., a body of knowledge.
  - How to prove theorems etc using this knowledge.
  - How to assimilate results to extend our knowledge.
- The body of knowledge should be
  - Efficient at proving theorems (relative entropy).
  - Parsimonious (entropy).
  - Structured (foundations, representation learning).
- HoTT foundations gives reasonable policies, values.

Learning to do Mathematics

- We must learn
  - Mathematics, i.e., a body of knowledge.
  - How to prove theorems etc using this knowledge.
  - How to assimilate results to extend our knowledge.
- The body of knowledge should be
  - Efficient at proving theorems (relative entropy).
  - Parsimonious (entropy).
  - Structured (foundations, representation learning).
- HoTT foundations gives reasonable policies, values.
- More structure than Chess, more depth than Go.

Tasks again

- Introduce/construct (invent, discover):

Tasks again

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, goals, (i.e.,terms, types), techniques, heuristics.

Tasks again

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, goals, (i.e.,terms, types), techniques, heuristics.
  - Recalling and using existing objects, including by analogy, generalization, instantiation (specialization).

Tasks again

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, goals, (i.e.,terms, types), techniques, heuristics.
  - Recalling and using existing objects, including by analogy, generalization, instantiation (specialization).
  - Experiment and judge plausibility.

Tasks again

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, goals, (i.e.,terms, types), techniques, heuristics.
  - Recalling and using existing objects, including by analogy, generalization, instantiation (specialization).
  - Experiment and judge plausibility.
- Evaluate (judge) based on:

Tasks again

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, goals, (i.e.,terms, types), techniques, heuristics.
  - Recalling and using existing objects, including by analogy, generalization, instantiation (specialization).
  - Experiment and judge plausibility.
- Evaluate (judge) based on:
  - Outcomes – known questions, simple statements with hard proofs, novelty, depth, applications etc.

Tasks again

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, goals, (i.e.,terms, types), techniques, heuristics.
  - Recalling and using existing objects, including by analogy, generalization, instantiation (specialization).
  - Experiment and judge plausibility.
- Evaluate (judge) based on:
  - Outcomes – known questions, simple statements with hard proofs, novelty, depth, applications etc.
  - Derived value – expected to be useful for outcomes.

Tasks again

- Introduce/construct (invent, discover):
  - Deductions, computations, proofs, solutions, backward deductions (e.g. case splitting), questions, goals, (i.e.,terms, types), techniques, heuristics.
  - Recalling and using existing objects, including by analogy, generalization, instantiation (specialization).
  - Experiment and judge plausibility.
- Evaluate (judge) based on:
  - Outcomes – known questions, simple statements with hard proofs, novelty, depth, applications etc.
  - Derived value – expected to be useful for outcomes.
- (Usually) depending on contexts and goals.

More tasks again

- Digest/Refine: both newly discovered mathematics and known mathematics (representation learning).

More tasks again

- Digest/Refine: both newly discovered mathematics and known mathematics (representation learning).
- Develop strategies for proving/solving : intermediate goals, allocate resources.

More tasks again

- Digest/Refine: both newly discovered mathematics and known mathematics (representation learning).
- Develop strategies for proving/solving : intermediate goals, allocate resources.
- Keep improving.

More tasks again

- Digest/Refine: both newly discovered mathematics and known mathematics (representation learning).
- Develop strategies for proving/solving : intermediate goals, allocate resources.
- Keep improving.
- Read and digest the literature (use NLP tools).

More tasks again

- Digest/Refine: both newly discovered mathematics and known mathematics (representation learning).
- Develop strategies for proving/solving : intermediate goals, allocate resources.
- Keep improving.
- Read and digest the literature (use NLP tools).
- Handle mathematics *in the large*.

More tasks again

- Digest/Refine: both newly discovered mathematics and known mathematics (representation learning).
- Develop strategies for proving/solving : intermediate goals, allocate resources.
- Keep improving.
- Read and digest the literature (use NLP tools).
- Handle mathematics *in the large*.
- Find good proofs – from which we can learn.

Conclusions

- AI systems in other fields have shown superhuman capabilities in many cognitive tasks.

- AI systems in other fields have shown superhuman capabilities in many cognitive tasks.
- For automating mathematics:

- AI systems in other fields have shown superhuman capabilities in many cognitive tasks.
- For automating mathematics:
    - clear approaches and workpoints.

- AI systems in other fields have shown superhuman capabilities in many cognitive tasks.
- For automating mathematics:
  - clear approaches and workpoints.
  - no evident barriers?

- AI systems in other fields have shown superhuman capabilities in many cognitive tasks.
- For automating mathematics:
  - clear approaches and workpoints.
  - no evident barriers?
- Partial progress towards the automating mathematics can lead to

- AI systems in other fields have shown superhuman capabilities in many cognitive tasks.
- For automating mathematics:
  - clear approaches and workpoints.
  - no evident barriers?
- Partial progress towards the automating mathematics can lead to
  - New uses of computers in discovering mathematics.

- AI systems in other fields have shown superhuman capabilities in many cognitive tasks.
- For automating mathematics:
  - clear approaches and workpoints.
  - no evident barriers?
- Partial progress towards the automating mathematics can lead to
  - New uses of computers in discovering mathematics.
  - Semantic search in the literature.

- AI systems in other fields have shown superhuman capabilities in many cognitive tasks.
- For automating mathematics:
  - clear approaches and workpoints.
  - no evident barriers?
- Partial progress towards the automating mathematics can lead to
  - New uses of computers in discovering mathematics.
  - Semantic search in the literature.
  - Automatic experimentation, testing, plotting, etc.

- AI systems in other fields have shown superhuman capabilities in many cognitive tasks.
- For automating mathematics:
  - clear approaches and workpoints.
  - no evident barriers?
- Partial progress towards the automating mathematics can lead to
  - New uses of computers in discovering mathematics.
  - Semantic search in the literature.
  - Automatic experimentation, testing, plotting, etc.
  - Search for objects with desired properties, combining various approaches.