If Mathematical Proof is a Game, What are the States and Moves?

David McAllester

AlphaGo Fan (October 2015)

AlphaGo Defeats Fan Hui, European Go Champion.

AlphaGo Lee (March 2016)

AlphaGo Zero vs. Alphago Lee (April 2017)

AlphaGo Lee:

- Trained on both human games and self play.
- Trained for Months.
- Run on many machines with 48 TPUs for Lee Sedol match.

AlphaGo Zero:

- Trained on self play only.
- Trained for 3 days.
- Run on one machine with 4 TPUs.
- Defeated AlphaGo Lee under match conditions 100 to 0.

AlphaZero Defeats Stockfish in Chess (December 2017)

AlphaGo Zero was a fundamental algorithmic advance for general RL.

The general RL algorithm of AlphaZero is essentially the same as that of AlphaGo Zero.

AlphaGo Zero

- The self-play training is based on completely new RL algorithm (described below).
- No rollouts are ever used.
- No database of human games is ever used.
- The deep networks are replaced with Resnet.
- A single dual-head network is used for both policy and value.

Training Time

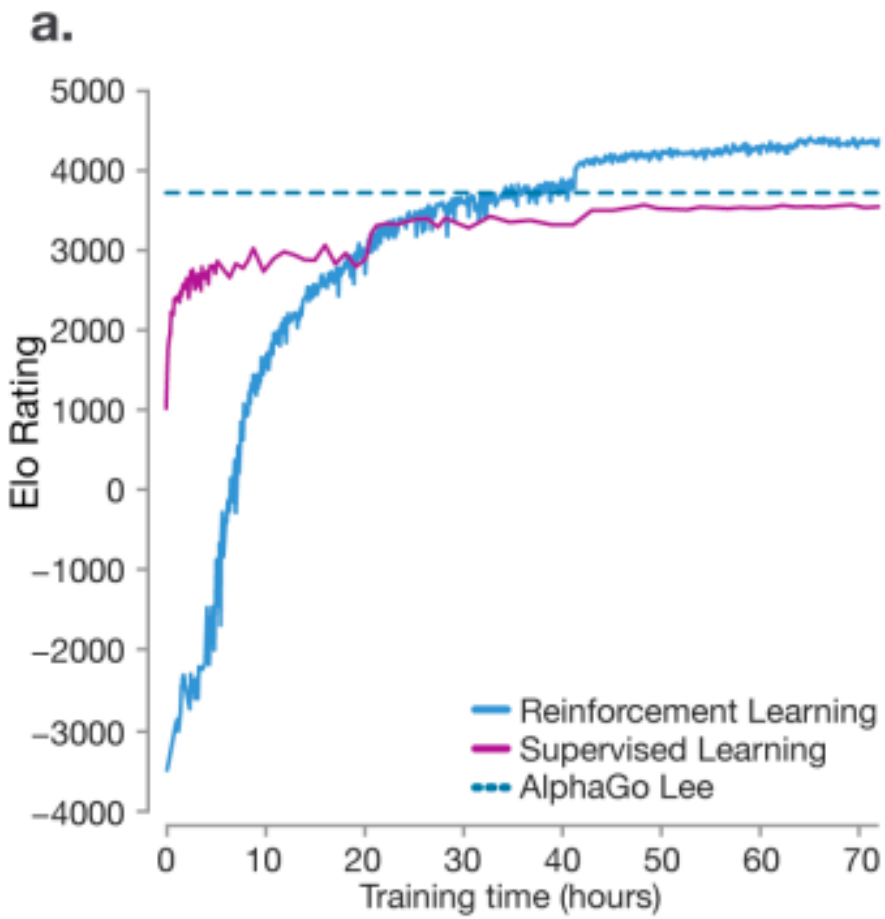
4.9 million games of self-play

0.4s thinking time per move

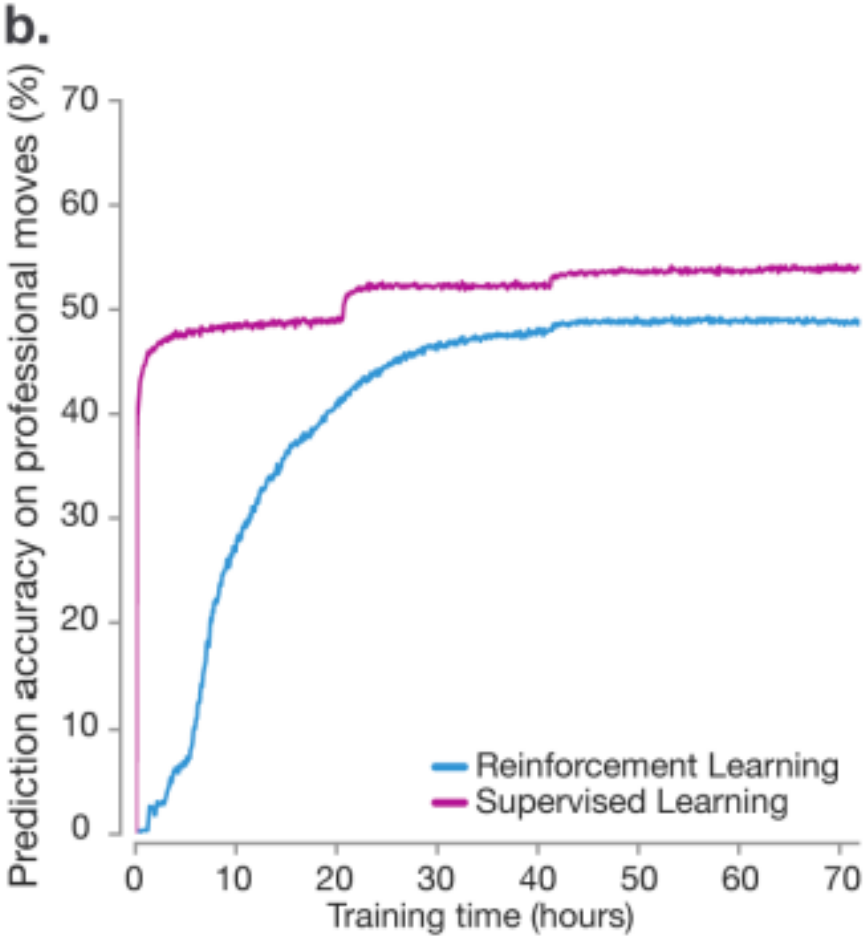
About 8 years of thinking time in training.

Training took just under 3 days — about 1000 fold parallelism.

Elo Learning Curve

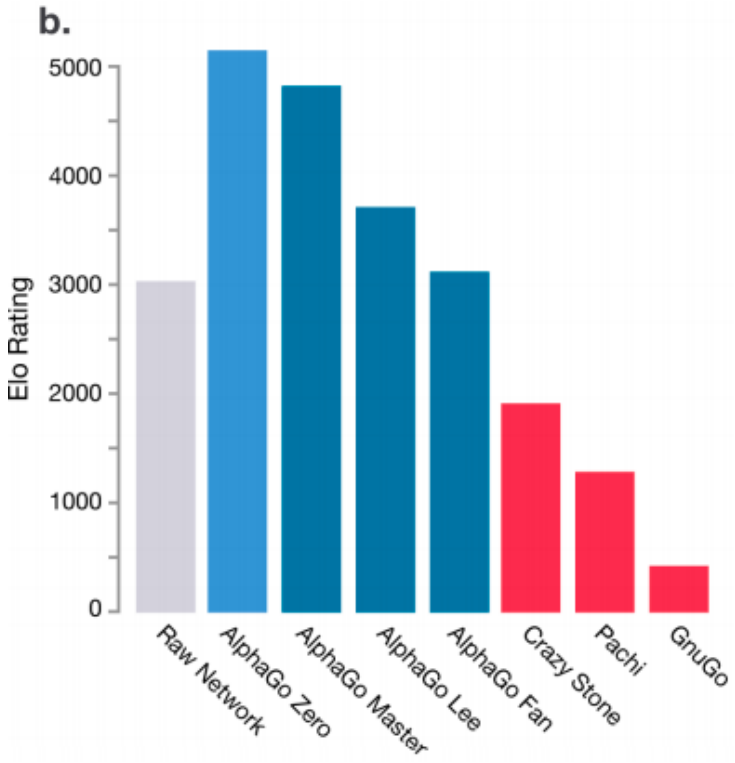


Learning Curve for Predicting Human Moves



Increasing Blocks and Training

Increasing the number of Resnet blocks form 20 to 40, and the number of training days from 3 to 40, gives an Elo rating over 5000.



AlphaZero Plays Chess

Essentially the same algorithm with the input image and output images modified to represent chess positions and move options respectively.

From white AlphaZero defeated Stockfish 25/50 and lost none.

From black AlphaZero won 3/50 and lost none.

Alpha evaluates 70 thousand positions per second.

Stockfish evaluates 80 million positions per second.

The New RL Algorithm — Tree-Search Bootstrapping

A neural network (a two-headed Resnet) provides both a (static) value and a stochastic policy (move probability).

These "static values" are used to guide a highly selective tree search.

The tree search produces a tree-derived position value and move probabilities.

The tree-values are used as data for training the static values.

More Specifically

UCT, a standard (2006) go algorithm, is used for tree search.

That this works for chess is shocking.

Each self-play game has a final outcome (win or loss) z.

For each position s reached in a self-play game we collect the data (s, π, z) where π is the tree-search-based move probability from s.

This data is collected in a replay buffer.

The Algorithm

Learning is done from this replay buffer using the following objective.

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \; E_{(s,\pi,z)\sim\text{Replay},\, a\sim\pi} \begin{pmatrix} (v_\Phi(s) - z)^2 \\ -\lambda_1 \log Q_\Phi(a|s) \\ +\lambda_2 ||\Phi||^2 \end{pmatrix}$$

Conspiracy Numbers

The unification of go and chess is surprising.

However, the original conspiracy numbers tree growth algorithm (McAllester, 1988) was designed for chess but bears a resemblance to UCT.

David Silver told me that they will try it.

Mathematics

Can one construct an artificial mathematician that learns entirely from "self play"?

What is "self-play" in open-domain mathematics?

I will consider the following principles.

- Mathematics is organized around concepts.
- Mathematics is driven by concept classification.

Mathematics is Organized Around Concepts

semigroups, groups, semirings, rings, fields,

vector spaces, Banach spaces, Hilbert spaces, differentiable manifolds, Lie groups, Lie algebras ...

strings, trees, graphs, relations, Kleene algebras

algebraic varieties, categories

Formalizing "Concept"

A concept can be formalized as a type expression.

Constructive Type Theory (HoTT)

ZFC-based type theory

Concepts

Concepts are like classes in programming languages. An instance is typically a tuple.

A group can be defined as a four-tuple $(S, \circ, \cdot^{-1}, 1)$ where

- S is the set of group elements
- $\circ$ is the group operation
- $\cdot^{-1}$ is the inverse operation on group elements
- 1 is the identity element

satisfying the group axioms.

"group" is a concept (a class).

Stereotypical Concepts and their Associated Isomorphisms

A stereotypical concept σ has instances which are pairs (S, a) where S is a "carrier set" and a is structure on that set — constants, functions, and predicates on S.

We have $(S, a) =_\sigma (U, b)$ if there exists a bijection from S to U that "carries" a to b.

When the structure on S (a and b) is defined by a simple type, the carrying operation can be defined by straightforward structural induction on simple type expressions.

Types vs. Formulas of Set Theory

We can define the class "group" as a formula $\Phi[x]$ of ZFC which is true if x is a group.

However we intuitively want the following substitution rule.

$$\frac{\Gamma;\ x : \mathbf{Group} \vdash \Phi[x] : \mathbf{Bool} \qquad \Gamma \vdash w =_{\mathbf{Group}} u}{\Gamma \vdash \Phi[w] \Leftrightarrow \Phi[u]}$$

ZFC-Based Type Theory

"ZFC-based" is taken to mean that the system defines the same set of theorems as ZFC — formal statements can be translated in either direction in natural a way preserving provability.

The translation from type theory to ZFC is defined by a natural set-theoretic semantics for type expressions.

The translation from ZFC to type theory is done using a natural concept of a Grothendieck Universe.

Mathematics is Driven by Concept Classification

The natural numbers are the isomorphism classes of "naked sets".

The ordinal numbers are the isomorphism classes of well-ordered sets.

The classification of simple finite groups.

The classification of compact two manifolds.

An A-Priori Distribution On Concepts

A concept is a closed type expression of dependent type theory (described below).

A distribution over concepts can be defined by a stochastic grammar.

Example:

$$\textbf{Function} \equiv \Sigma_{s:\textbf{Set}}\ s \to s$$

The concepts of semigroup, group, ring and field should all be accessible under random sampling.

A Mathematics Game

Maintain a database of concepts.

Repeat:

- Draw a concept σ from some time-evolving distribution.
- Work (for some time) on the classification of σ.

The evolving concept distribution is, of course, an important issue.

Classifying a Concept σ

- Can we find $f : \tau \to \sigma$ generating inhabitants of σ? For example, the free group over a set of generators.
- Can we find $g : \sigma \to \tau$ defining σ-invariants. For example, the cardinality of a finite set, the parity of a permutation, the fundamental group of a topological space.
- If we can find a concept τ with $f : \tau \to \sigma$ and $g : \sigma \to \tau$, with f and g establishing a bijection, then σ and τ are cryptomorphic and should be merged.

All of the above "functors" must be "natural".

Starting from "set"

The natural numbers arise as the isomorphism classes of the finite sets.

Addition arises as disjoint union and multiplication arises as cross product.

The integers arise by extending the natural numbers to a group.

The rational numbers arise by extending the integers to a field.

Vector spaces might arise as a generalization of $\mathbb{Q}^2$.

The real numbers might arise as the completion of the rationals (requires completion as an operation on metric spaces).

The complex numbers?

Type Theory Details — dependent Pair Types

Tuples can be built form pairs — a triple (x, y, z) can be represented by $(x, (y, z))$.

The type of pairs (x, y) with $x \in \sigma$ and $y \in \tau[x]$ is written (perversely) as $\Sigma_{x:\sigma}\, \tau[x]$.

For example, the class of "pointed sets" (S, a) with S a set and $a \in S$ is written as $\Sigma_{S:\mathbf{Set}}\, S$.

We write $\sigma \times \tau$ for $\Sigma_{x:\sigma}\, \tau$ where x does appear in τ.

Formulas

Atomic formulas:

- $P(e)$ with $e \in \sigma$ and $P \in (\sigma \to \mathbf{Bool})$.
- set-theoretic equalities $e_1 \doteq e_2$
- isomorphism equalities $e_1 =_\sigma e_2$

Boolean and quantified formulas $\neg\Phi$, $\Phi_1 \vee \Phi_2$ and $\forall x : \sigma\ \Phi[x]$.

Groups

A Group can also be defined as a pair $(S, \circ)$.

$$\mathbf{Magma} \equiv \Sigma_{S:\mathbf{Set}}\, S \times S \to S$$

$$\mathbf{Group} \equiv S_{G:\mathbf{Magma}}\, \Phi[G]$$

In general $S_{x:\sigma}\, \Phi[x]$ is the subclass of $x \in \sigma$ such that $\Phi[x]$.

The Full System

variables, pairs	x	(e_1, e_2)	$\pi_i(e)$
functions	$\lambda x\!:\!\sigma\ e[x]$	$f(e)$	
Booleans	$P(e)$	$e_1 \doteq e_2$	$e_1 =_\sigma e_2$
	$\neg\Phi$	$\Phi_1 \vee \Phi_2$	$\forall x\!:\!\sigma\ \Phi[x]$
types	$\Sigma_{x:\sigma}\ \tau[x]$	$\Pi_{x:\sigma}\ \tau[x]$	$S_{x:\sigma}\ \Phi[x]$
	Bool	**Set**	**Class**

Deriving Isomorphism

We have

$$(s, a) =_{\Sigma_{\alpha:\mathbf{Set}}\ \tau[\alpha]} \quad (u, b)$$

if there exists a bijection $f : \ u \to v$ which "carries" a to b.

$$\Gamma \vdash u, v : \mathbf{Set}, f : \mathbf{Bijection}[u, v]$$
$$\Gamma;\ \alpha : \mathbf{Set} \vdash \tau[\alpha] : \mathbf{Set}$$

$$\Gamma \vdash \begin{array}{l} \forall\, h : \tau[u] \\ \quad (u, h) \ =_{\Sigma_{\alpha:\mathbf{Set}}\ \tau[\alpha]} \ (v, \mathbf{Carrier}(u, v, f, (\lambda\, \alpha : \mathbf{Set}\ \tau[\alpha]))(h)) \end{array}$$

The Substitution of Isomorphics

$$\Gamma \vdash \sigma, \tau : \mathbf{Class}$$

$$\frac{\Gamma;\ x : \sigma \vdash e[x] : \tau \qquad \Gamma \vdash w =_\sigma u}{\Gamma \vdash e[w] =_\tau e[u]}$$

Summary I

AlphaZero embodies a new power machine learning algorithm based on tree-search bootstrapping.

Tree-search bootstrapping seems very well suited to learning to prove theorems.

This leads to question of whether a computer could become a super-human mathematician through "self-play" in open-domain mathematics.

Summary II

But how do we formulate the objective of open-domain mathematics?

Open-domain mathematics is organized around concepts definable in type theory.

A possible objective for open-domain mathematics is to classify concepts under some (evolving) concept distribution.

END