AGI via Combining Logic with Deep Learning

甄景贤 (King-Yin Yan)

General.Intelligence@Gmail.com

Abstract. An integration of deep learning and logic is proposed, based on the Curry-Howard isomorphism. Under this interpretation, it turns out that Google's BERT, which many currently state-of-the-art language models are derived from, can be regarded as a special form of logic. Moreover, this structure can be incorporated under a reinforcement-learning framework to form a minimal AGI architecture. We also mention some insights gleaned from category and topos theory that may be helpful to practitioners of AGI.

Keywords: deep learning, symbolic logic, logic-based AI, neural-symbolic integration, Curry-Howard isomorphism, category theory, topos theory, fuzzy logic

0 Introduction

The results in the present paper does not make use of category theory in any significant way (nor the Curry-Howard isomorphism, for that matter). Its main accomplishment is to express AGI in the categorical language. To the lay person the concepts of category theory (such as pullbacks, adjunctions, toposes, sheaves, ...) may be difficult to grasp, but they are the mathematician's "daily bread". We hope that describing AGI in categorical terms will entice more mathematicians to work on this important topic.

Secondly, an abstract formulation allows us to see clearly what is meant by "the mathematical structure of logic", without which logic is just a collection of strange rules and axioms, leaving us with a feeling that something may be "amiss" in our theory.

0.1 The Curry-Howard Isomorphism

As the risk of sounding too elementary, we would go over some basic background knowledge, that may help those readers who are unfamiliar with this area of mathematics.

The Curry-Howard isomorphism expresses a connection between logic **syntax** and its underlying **proof** mechanism. Consider the mathematical declaration of a **function** f with its domain and co-domain:

$$f : A \to B. \tag{1}$$

This notation comes from type theory, where A and B are **types** (which we can think of as sets or general spaces) and the function f is an **element** in the function space $A \to B$, which is also a type.

What the Curry-Howard isomorphism says is that we can regard $A \to B$ as a **logic** formula $A \Rightarrow B$ (an implication), and the function f as a **proof** process that maps a proof of A to a proof of B.

The following may give a clearer picture:

$$\begin{array}{ll} \boxed{\text{logic}} & A \Longrightarrow B \\ & \text{- - - - - - - - - -} \\ \boxed{\text{program}} & \blacksquare \overset{f}{\longmapsto} \blacksquare \quad . \end{array} \tag{2}$$

What we see here is a logic formula "on the surface", with an underlying proof mechanism which is a **function**. Here the ■'s represent proof objects or witnesses. The logic propositions A and B coincide with the **domains** (or **types**) specified by type theory. Hence the great educator Philip Wadler calls it "propositions as types". [1] Other textbooks on the Curry-Howard isomorphism include: [29] [28] [31].

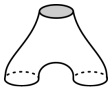
The gist of our theory is that Deep Learning provides us with neural networks (ie. non-linear functions) that serve as the proof mechanism of logic via the Curry-Howard isomorphism. With this interpretation, we can impose the

[1] See his introductory video: https://www.youtube.com/watch?v=IOiZatlZtGU .

mathematical structure of logic (eg. symmetries) onto neural networks. Such constraints serve as **inductive bias** that can accelerate learning, according to the celebrated "No Free Lunch" theory [34].

In particular, logic propositions in a conjunction (such as $A \wedge B$) are commutative, ie. invariant under permutations, which is a "symmetry" of logic. This symmetry essentially decomposes a logic "state" into a set of propositions, and seems to be a fundamental feature of most logics known to humans. Imposing this symmetry on neural networks gives rise to symmetric neural networks, which can be easily implemented thanks to a separate result by other researchers. This is discussed in §3.

As an aside, the Curry-Howard isomorphism also establishes connections to diverse disciplines. Whenever there is a space of elements and some operations over them, there is a chance that it has an underlying "logic" to it (see eg. Baez and Stay's "Rosetta Stone" paper: [1], also [8]). For example, in quantum mechanics, that of Hilbert space and Hermitian operators. Another example: in String Theory, strings and cobordisms between them (The following is the famous "pair of pants" cobordism, representing a process in time that merges two strings into one (time is read upwards)):



(3)

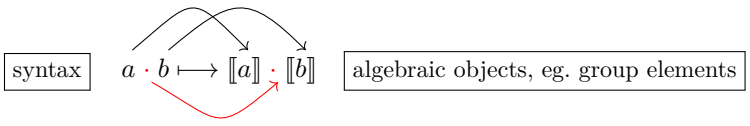
Seeing logical types as topological spaces is also the origin of Voevodsky's **Homotopy Type Theory** (HoTT) [20], where the identity of two inhabitants in a type is seen as a homotopy **path**.

1 Prior Research

1.1 Neuro-Symbolic Integration

There has been a long history of attempts to integrate symbolic logic with neural processing, with pioneers such as Ron Sun, Dov Gabbay, Barbara Hammer, among others. We consider two **model-based** approaches below.

From a categorical perspective, model theory is a **functor** mapping logic syntax to algebraic objects and operations between them (hence the name "functorial semantics"):



(4)

Model theory is interesting when the target structure has additional properties not specified by the logic syntax. For example, the predicate `male(x)` may be modelled by:

algebraic geometry	`male`(x) $\Leftrightarrow f(x) \geq 0$	f is a polynomial
linear algebra	`male`(x) $\Leftrightarrow Mx \geq 0$	M is a matrix
topology	`male`(x) $\Leftrightarrow x \in S$	S is an open set

(5)

Model-based methods appear to be impractical for AGI because the number of grounded atomic propositions would be too large (potentially infinite, if we include also propositions that are imagined). However, if all possible atoms are embedded in a mathematical space through mapping schemes such as the above (5), it may be feasible.

In the **type-theoretic** or "syntactic" approach (including the one in this paper), propositions exist in the product space of predicates and objects; For example the predicate $P(a, b)$ lives in the space $\mathbb{P}\text{red} \times \mathbb{O}\text{bj} \times \mathbb{O}\text{bj}$ where $\mathbb{P}\text{red}$ is the space of all possible predicates and $\mathbb{O}\text{bj}$ the space of all possible objects [2]. **Inference** is performed by the main neural network, while **learning** changes the network weights. This is simple and straightforward.

Whereas, in the **model-theoretic** approach one places objects in a high dimensional space such that their positions satisfy the constraints imposed by various predicates (eg. polynomials or matrices or open sets). Now **inference** occurs as the system pays *attention* to some points in an $\mathbb{O}$bject space, which points are covered by some predicates, thus forming propositions, leading to awareness of new propositions, ... and so on. It is interesting that, under this scheme, it seems as if all truths are known *a priori*, and the system just needs to pay attention to them. **Learning** changes the geometric shapes of predicates and thus forms new truths to be discovered by the system.

[2] Here objects mean logical or first-order objects, not categorical objects

1. In Pascal Hitzler and Anthony Seda's **Core Method** [10], an **interpretation** $\mathcal{I}$ is a function that assigns truth values to the set of all possible ground atoms in a logic language $\mathcal{L}$. One can see $\mathcal{I}$ as an enumeration of ground atoms that are true, and thus it provides a model to interpret any logic formula in $\mathcal{L}$. Moreover $\mathcal{I}$ is a function from the space X of atoms to $\mathbf{2} = \{\top, \bot\}$ and can be given a topology $\mathbf{2}^X$ which is X copies of the discrete topology of $\mathbf{2}$. Such a topology makes $\mathcal{I}$ homeomorphic to the **Cantor set** in $[0, 1]$. To a logic program P is associated a **semantic operator** $\mathcal{T}_P : \mathcal{I} \to \mathcal{I}$, performing a single step of forward **inference**. Finally, the space of interpretations $\mathcal{I}$ is embedded into $\mathbb{R}$ using a "level mapping" (the level of an atom increases by each inference step; All the atoms of an interpretation $\mathcal{I}$ are translated into a fractional number in base b). This allows $\mathcal{T}_P$ to be approximated by a neural network $f : \mathcal{I} \to \mathbb{R}$.
   The goal of this research is to find the fixed-point semantics of logic programs, but with suitable modifications, the same mathematical structure may be used to build an inference engine or AGI. In such case, the logic program would function as the **knowledge base** while interpretations would play the role of **working memory** (though the memory could only be a subset of an interpretation, due to physical limitation).
2. **∂-ILP** [5] is focused on the learning problem. A **valuation** is a vector $[0, 1]^n$ mapping every ground atom to a real number $\in [0, 1]$. Each clause is attached with a Boolean flag to indicate whether it is included in the results or not. From each clause c one can generate a function $\mathcal{F}_c$ on valuations that implements a single step of forward **inference**. To enable differentiability, the Boolean flag is relaxed to be a continuous value and gradient descent is used to **learn** which clauses should be included.

We would also like to mention Geoffrey Hinton's recent **GLOM theory** [9], which addresses the problem of representing a hierarchy of visual structures. This further supports that representing and learning **relational** (logical) knowledge is a research topic of central importance, and that there is a convergence of "mainstream" AI with AGI.

1.2 Cognitive Architectures and Reinforcement Learning

Reinforcement Learning (RL). In the 1980's, Richard Sutton [30] introduced reinforcement learning as an AI paradigm, drawing inspiration from Control Theory and Dynamic Programming. In retrospect, RL already has sufficient generality to be considered an AGI theory, or at least as a top-level framework for describing AGI architectures.

Relation to AIXI. AIXI is an abstract AGI model introduced by Marcus Hutter in 2000 [13]. AIXI's environmental setting is the external "world" as observed by some sensors. The agent's internal model is a universal Turing machine (UTM), and the optimal action is chosen by maximizing potential rewards over all programs of the UTM. In our (minimal) model, the UTM is *constrained* to be a neural network, where the NN's **state** is analogous to the UTM's **tape**, and the optimal weights (program) are found via Bellman optimality.

Relation to Quantum mechanics and Path Integrals. At the core of RL is the Bellman equation, which governs the update of the utility function to reach its optimal value. This equation (in discrete time) is equivalent to the Hamilton-Jacobi equation in differential form. Nowadays they are unified as the Hamilton-Jacobi-Bellman equation, under the name "optimal control theory" [18]. In turn, the Hamilton-Jacobi equation is closely related to the Schrödinger equation in quantum mechanics:

$$\boxed{\text{Bellman eqn.}} \leftrightsquigarrow \boxed{\text{Hamilton-Jacobi eqn.}} \leftrightsquigarrow \boxed{\text{Schrödinger eqn.}} \tag{6}$$

but the second link is merely "heuristic"; it is the well-studied "quantization" process whose meaning remains mysterious to this day. Nevertheless, the path integral method introduced by Richard Feynmann can be applied to RL algorithms, eg. [16].

The Hamilton-Jacobi equation gives the RL setting a "symplectic" structure [19]; Such problems are best solved by so-called symplectic integrators (proposed by 冯康 (Feng Kang) in the 1980s [6], see also [17]). Surprisingly, in the RL / AI literature, which has witnessed tremendous growth in recent years, there is scarcely any mention of the Hamilton-Jacobi structure, while the most efficient heuristics (such as policy gradient, experience replay, Actor-Critic, etc.) seem to exploit other structural characteristics of "the world".

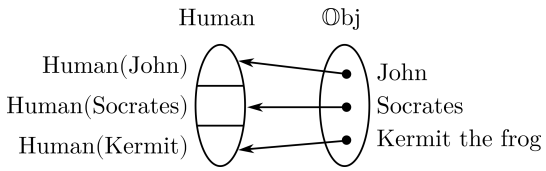
2 The Mathematical Structure of Logic

Currently, the most mathematically advanced and satisfactory description of logic seems to base on category theory, known as categorial logic and topos theory. This direction was pioneered by William Lawvere in the 1950-60's. The body of work in this field is quite vast, but we shall briefly mention some points that are relevant to AGI. A more detailed tutorial on categorical logic, with a focus on AGI, is in preparation [35].

2.1 Predicates and Dependent Type Theory

The Curry-Howard isomorphism identifies *propositional* intuitionistic logic with type theory. As such, the arrow $\to$ in type theory is "used up" (it corresponds to the implication arrow $\Rightarrow$ in intuitionistic logic). However, predicates are also a kind of functions (arrows), so how could we accomodate predicates in type theory such that Curry-Howard continues to hold? This is the idea behind Martin Löf's **dependent type theory**.

In dependent type theory, a predicate $P(\cdot)$ is a **type constructor** ([29] §8.7) taking an element a of type $\mathbb{O}\text{bj}$ to create a new type $P(a)$. For example, each element $a \in \{\text{John, Socrates, Kermit}\}$ creates a new type Human(a), and thus Human is a **family** of types or a **dependent type**:



(7)

Mathematically, a dependent type is a **product** of types indexed by another type, denoted $\Pi_A B$. If every source element maps to the same type B, then this product *degenerates* into the ordinary function type $A \to B$.

An immediate question is: what kind of structure does dependent type theory impose on the inference operator $\vdash$, and thus our neural network? The answer is surprising: almost none. Curry-Howard isomorphism merely tells us that predicates such as $P(\cdot)$ is a **type constructor** taking an existing type to create a new type, and an atomic proposition such as $P(a)$ would be a **dependent type**. It does not constrain the functions mapping between such spaces.

The expressiveness of predicate logic (in one form or another) is a highly desirable feature for AGI knowledge representations. So it seems necessary to incorporate dependent type theory into our logic. From a categorical perspective, predicates can be regarded as **fibers** over a base set; This is the treatment given in Bart Jacob's book [14]. Thus category theory gives us more insight into the (predicate) structure of logic, though it is as yet unclear how to make use of this particular idea.

2.2 (Fuzzy) Topos Theory

The author's previous paper [36], almost a decade ago, proposed a fuzzy-probabilistic logic where probabilities are distributed over fuzzy truth values. So far we still believe that regarding fuzziness as a generalization of binary truth is philosophically sound. Thus it behooves to develop a generalization of standard topos theory to the fuzzy case.

The most important commutative diagram in Topos theory is this one:

$$\begin{array}{ccc} X & \xrightarrow{!} & 1 \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle \text{true}} \\ Y & \xrightarrow[\chi_m]{} & \Omega \end{array} \tag{8}$$

It can be understood as saying that every **set** is a **pullback** of the true map $1 \to \Omega$ (which "picks out" true from $\Omega = \{\top, \bot\}$), in analogy to the idea of a "moduli space" where every family is a pullback of a "universal family" [25] [7]. Following this idea, could it be that every fuzzy set is the pullback of a fuzzy "true" map?

The book [2] §5.2.4 provides a concise review of the categorical treatment of fuzzy sets: The sub-object classifier Ω that characterizes classical set theory is generalized to a **complete Heyting algebra** (CHA, also called a **frame**, which captures the structure of a topology, ie, the lattice of open subsets of a set; This includes the interval $[0, 1]$ as a special case), and thus leads to the recognition that *the internal logic of a topos is intuitionistic*.

This line of research culminated in Höhle's [11] and [12], where fuzzy set theory is interpreted as sub-fields of sheave theory, ie, complete Ω-valued sets, where Ω is a frame. More recent papers seem to support this thinking [15] [33].

3 Permutation Symmetry and Symmetric Neural Networks

From the categorical perspective, we make the following correspondence with logic and type theory:

$$\begin{array}{lcccl} \boxed{\text{product}} & A \times B & \leftrightsquigarrow & A \wedge B & \boxed{\text{conjunction}} \\ \boxed{\text{function}} & A \to B & \leftrightsquigarrow & A \Rightarrow B & \boxed{\text{implication}}. \end{array} \tag{9}$$

One basic characteristic of (classical) logic is that the conjunction $\wedge$ is **commutative**:

$$\mathsf{P} \wedge \mathsf{Q} \quad \Leftrightarrow \quad \mathsf{Q} \wedge \mathsf{P}. \tag{10}$$

This remains true of probabilistic logic, where $\wedge$ and $\vee$ are unified as conditional probability tables (CPTs) of the nodes of Bayesian networks.

Once we know the symmetry, the question is how to impose this symmetry on deep neural networks. Interestingly, the answer already comes from an independent line of research (namely, PointNet [22] and Deep Sets [37]) that deals with visual object recognition of point clouds, eg:

(11)

In a point cloud, it does not matter the order in which the points are presented, as inputs to the classifier function. Such a function needs to be permutation invariant to a huge number of points.

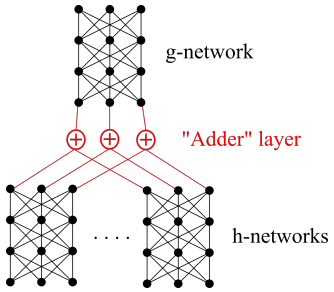
From [37]: the **Kolmogorov–Arnold representation theorem** states that every multivariate continuous function can be represented as a sum of continuous functions of one variable:

$$f(x_1, ..., x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right) \tag{12}$$

It can be specialized to such that every symmetric multivariate function can be represented as a sum of (the same) functions of one variable:

$$f(x_1, ..., x_n) = g(h(x_1) + ... + h(x_n)) \tag{13}$$

This leads to the following implementation using neural networks:



(14)

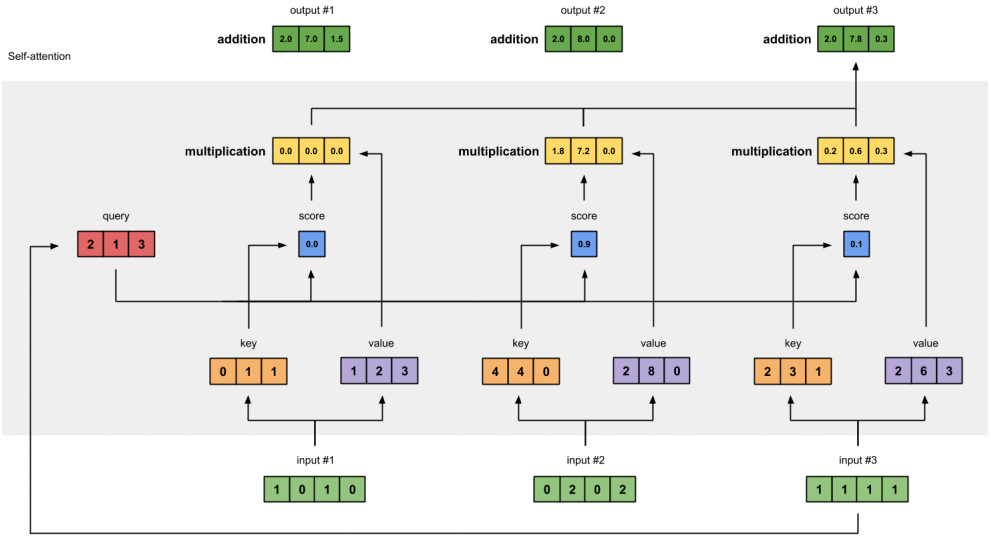
And this can be easily implemented with a few lines of Tensorflow, see §5.

3.1 Why BERT is a Logic

BERT (and its variants) are based on the Transformer architecture [4], and Transformers are based solely on the Self-Attention mechanism [32]. In the following diagram [3], one can verify that the Transformer is permutation-invariant (or more precisely, **equivariant**). That is to say, for example, if input #1 and #2 are swapped, then output #1 and

[3] From blog article: Illustrated: Self-Attention – Step-by-step guide to self-attention with illustrations and code https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

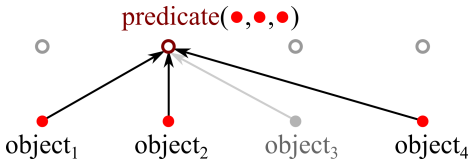
#2 would also be swapped:



(15)

In other words, each Transformer layer takes N inputs and produces N equivariant outputs. That is the same as saying that *each* output is permutation-invariant in all its inputs. As we explained in the last section, permutation invariance is the symmetry that characterizes a logic as having *individual* propositions.

The following is a simplified diagram of an Attention layer. The output is a new proposition that depends on the input objects, and thus, functions as a **predicate**:



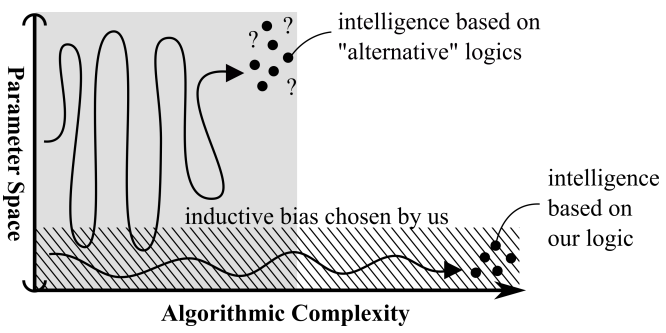
(16)

In **Multi-Head Attention**, the intermediate computations are duplicated multiple (eg, $M = 8$) times, each with their own weight matrices. From the logic point of view, this amounts to duplicating M logic rules per output. But since the next layer still expects N inputs, the M outputs are combined into one, before the next stage. Thus, from the logic point of view this merely increased the parameters *within* a single logic rule, and seems not significant to increase the power of the logic rule-base. Indeed, experimental results seem to confirm that multi-head attention is not particularly gainful towards performance.

A comment is in order here, about the choice of the word "head". In logic programming (eg Prolog), one calls the conclusion of a logic rule its "head", such as `P` in `P :- Q,R,S`. Perhaps the creators of BERT might have logic rules in mind?

4 "No Free Lunch" Theory

The following conceptual diagram of the algorithmic **search space** illustrates the possibility that there might exist some form of logic that is drastically different from the symbolic logic currently known to humans:



(17)

but there is no efficient algorithm to find them (grey area is much larger than shaded area). The permutation symmetry proposed in this paper forces our logic to be decomposable into **propositions**. Such a logical form allows a mental state to be enumerated as a list of sentences (propositions), same as the "linear" structure of human **languages**. If the AGI knowledge representation is linear (in the sequential sense) and symbolic, then it would not be far from our formulation – all these logics belong to one big family.

But could there be drastically different logics? One observes that pictures and music are not easily described by words, indeed they are 2-dimensional structures. This suggests that the brain may use **multi-dimensional** arrays of features to represent the world. Such a "logic" would be very different from sequential logic and it would be interesting and fruitful to analyze the relation between them.

5 Experiment

A simple test [4] of the symmetric neural network, under reinforcement learning (Policy Gradient [5]), has been applied to the Tic-Tac-Toe game.

The state of the game is represented as a set of 9 propositions, where all propositions are initialized as "null" in the beginning. During each step of the game, a new proposition is added to the set (ie. over-writing the null propositions). Each proposition encodes who the player is, and which square (i, j) she has chosen. In other words, it is a predicate of the form: `move(player,i,j)`. The neural network takes 9 propositions as input, and outputs a new proposition; Thus it is a permutation-invariant function.

In comparison, the game state of traditional RL algorithms (eg. AlphaGo [26] [27] [21]) usually is represented as a "chessboard" vector (eg. 3×3 in Tic-Tac-Toe, 8×8 in Chess, $19 \times 19 = 361$ in Go [6]). This state vector is the same constant length even if there are very few pieces on the chessboard. Our logic-based representation may offer some advantages over the board-vector representation, and likely induces a different "way of reasoning" about the game.

In our Tic-Tac-Toe experiment, convergence of learning is observed, but the algorithm fell short of achieving the highest score (19 instead of 20), and the score displayed unstable oscillating behavior after it got near the optimal value. In comparison, the board-vector version (also using policy gradient) fails to get near the highest score after 2 hours, and it also shows oscillatory behavior. Further investigation is required, but this seems to be a promising start.

6 Conclusion and Future Directions

We described a minimal AGI with a logic that can derive one new proposition per iteration. This seems sufficient to solve simple logic problems such as Tic-Tac-Toe. As a next step, we would consider inference rules with multi-proposition conclusions. The latter seems essential to **abductive** reasoning. For example, one can deduce the concept "apple" from an array of visual features; Conversely, the idea of an "apple" could also evoke in the mind a multitude of features, such as color, texture, taste, and the facts such as that it is edible, is a fruit, and that Alan Turing died from eating a poisoned apple (a form of episodic memory recall), and so on. This many-to-many inference bears some similarity to the brain's computational mechanisms [24] [23] [3]. The author is embarking on an abstract unifying AGI theory that makes references to (but not necessarily copying) brain mechanisms.

Acknowledgements

Thanks Ben Goertzel for suggesting that neural networks are advantageous over pure symbolic logic because they have fast learning algorithms (by gradient descent). That was at a time when "deep learning" was not yet a popular word. Thanks Dmitri Tkatch for pointing me to existing research of symmetric neural networks. Thanks Dr. 肖达 (Da Xiao) for explaining to me details of BERT.

[4] Code with documentation is on GitHub: https://github.com/Cybernetic1/policy-gradient

[5] The Policy Gradient algorithm is chosen because it allows *continuous* actions. Other reinforcement learning algorithms require learning the value function over actions, and when the action space is not discrete such a value function cannot be represented by a table, but perhaps as a neural network. However, it is not easy to find the *maximum* of a neural network, which is required to choose the optimal action. Policy Gradient avoids this because the policy function directly maps to actions.

[6] In AlphaGo and AlphaZero, the algorithm makes use of several auxiliary "feature planes" that are also chessboard vectors, to indicate which stones have "liberty", "ko", etc.

Also thanks to the following people for invaluable discussions over many years: Ben Goertzel, Pei Wang (王培), Abram Demski, Russell Wallace, Juan Carlos Kuri Pinto, SeH, Jonathan Yan, and others. Also thanks to all the university professors and researchers in Hong Kong (especially in the math departments, and their guests), strangers who taught me things on Zhihu.com (知乎), Quora.com, and StackOverflow.

References

[1] John Baez and Mike Stay. "Physics, topology, logic and computation: a Rosetta Stone". In: *New structures for physics*. Springer, 2010, pp. 95–172.
[2] Bělohlávek, Dauben, and Klir. *Fuzzy logic and mathematics: a historical perspective*. Oxford University Press, 2017.
[3] Thomas Boraud. *How the brain makes decisions*. Oxford, 2020.
[4] Devlin et al. "BERT: pre-training of deep bidirectional transformers for language understanding". In: (2018). URL: arXiv:%201810.04805v2%20[cs.CL].
[5] Evans and Grefenstette. "Learning explanatory rules from noisy data". In: *Journal of artificial intelligence research* (2017).
[6] Kang Feng and Mengzhao Qin. *Symplectic geometric algorithms for Hamiltonian systems*. Springer, 2010.
[7] Joe Harris and Ian Morrison. *Moduli of curves*. Vol. 187. Springer Science & Business Media, 2006.
[8] Chris Heunen and Jamie Vicary. *Categories for Quantum Theory: an introduction*. Oxford University Press, 2019.
[9] Geoffrey Hinton. "How to represent part-whole hierarchies in a neural network". In: *arXiv preprint arXiv:2102.12627* (2021).
[10] Hitzler and Seda. *Mathematical aspects of logic programming semantics*. CRC Press, 2011.
[11] Ulrich Höhle. "Fuzzy sets and sheaves. Part I: basic concepts". In: *Fuzzy Sets and Systems* 158.11 (2007), pp. 1143–1174.
[12] Ulrich Höhle. "Fuzzy sets and sheaves. Part II:: sheaf-theoretic foundations of fuzzy set theory with applications to algebra and topology". In: *Fuzzy Sets and Systems* 158.11 (2007), pp. 1175–1212.
[13] Marcus Hutter. *Universal artificial intelligence*. Springer, 2005.
[14] Bart Jacobs. *Categorical logic and type theory*. Elsevier, 1999.
[15] Jardine. "Fuzzy sets and presheaves". In: (2019). URL: arXiv:%201904.10314v5%20[math.CT].
[16] Kappen. "An introduction to stochastic control theory, path integrals and reinforcement learning". In: *AIP conference proceedings 887* 149 (2007). URL: https://doi.org/10.1063/1.2709596.
[17] Benedict Leimkuhler and Sebastian Reich. *Simulating hamiltonian dynamics*. 14. Cambridge university press, 2004.
[18] Daniel Liberzon. *Calculus of variations and optimal control theory: a concise introduction*. Princeton Univ Press, 2012.
[19] Peter Mann. *Lagrangian and Hamiltonian dynamics*. Oxford University Press, 2018.
[20] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. 2013. arXiv: 1308.0729 [math.LO].
[21] Pumperla and Ferguson. *Deep learning and the game of Go*. Manning, 2019.
[22] Qi et al. "Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *CVPR* (2017). https://arxiv.org/abs/1612.00593.
[23] Edmund Rolls. *Brain computation – what and how*. Oxford, 2021.
[24] Edmund Rolls. *Cerebral cortex – principles of operation*. Oxford, 2016.
[25] Martin Schlichenmaier. *An introduction to Riemann surfaces, algebraic curves and moduli spaces*. Springer Science & Business Media, 2010.
[26] et al. Silver. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* (529 2016), pp. 484–489.
[27] et al. Silver. "Mastering the game of Go without human knowledge". In: *Nature* (550 2017), pp. 354–359.
[28] Simmons. *Derivation and computation: taking the Curry-Howard correspondence seriously*. Cambridge University Press, 2000.
[29] Sørensen and Urzyczyn. *Lectures on the Curry-Howard isomorphism*. Elsevier, 2006.
[30] Sutton. "Temperal credit assignment in reinforcement learning". University of Massachusetts, Amherst, 1984.
[31] Thompson. *Type theory and functional programming*. Addison-Wesley, 1991.
[32] Vaswani et al. "Attention is all you need". In: (2017). https://arxiv.org/abs/1706.03762.
[33] Vickers. "Fuzzy sets and geometric logic". In: *Fuzzy sets and systems* 161 (2010), pp. 1175–1204.
[34] Wolpert and Macready. "No free lunch theorems for optimization". In: *IEEE transactions on evolutionary computation 1* 67 (1997).
[35] Yan. *AGI logic tutorial*. 2021. URL: https://drive.google.com/file/d/1v2efrH4gVJS9wG-KKgi1uFbbCoMOc9Hl/view?usp=sharing.
[36] King-Yin Yan. "Fuzzy-probabilistic logic for common sense reasoning". In: *Artificial general intelligence 5th international conference, LNCS 7716* (2012).

[37] Zaheer et al. “Deep Sets”. In: *NIPS 2017* (2017). https://arxiv.org/abs/1611.04500.

(To the editor: Diagrams in this paper can be re-organized in the traditional format, please contact the author if this is desired.)