Some basic notions in AGI & my theory

YKY 甄景贤

Independent researcher, Hong Kong

generic.intelligence@gmail.com

May 15, 2019

Talk summary

1. What is inductive bias? “No free lunch” theorem
2. What gives neural networks their power?
3. Turing machines and universal logic
4. Structure of classical AI systems

Section 1

What is inductive bias? “No free lunch” theorem

The goal of machine learning

- The goal of machine learning is to search in a **space** of learning machines, those machines that satisfy certain criteria
- For example, among the neural networks of a certain size and shape, find the weights that satisfy an **objective function**

AI Winter

- Generally speaking, bottleneck problem of AI = **search space too large**, thus learning too slow
- Historically, "AI Winter" occurred because **logic-based** AI learning suffers from combinatorial explosion, and we lacked workable **heuristics** to tackle it

Inductive bias

- Every learning algorithm has its **inductive bias**
- That is to say, some regions of the search space would **not** be searched
- Bias makes learning faster
- But if bias is too strong, the space containing the solution would be cut off "Throw the baby out with the water"

"No free lunch" theorem

- When search space has no *a priori* structure, any inductive bias would be **good at some problems while bad at others** — "no free lunch"
- For example, vision has the structure of 3D Euclidean geometry, thus the human visual cortex may have inductive bias for this invariance
- Or, human cognition has "logical" structure, using this inductive bias may accelerate machine learning of human intelligence

Kolmogorov complexity

- is **incomputable**, but **approximable**
- The **semantic distance metric** between logic propositions is related to it, where one logic deduction step corresponds to 1 unit of semantic distance
- Find a set of logic rules, that **explains** the world, and not deduce false facts, and # of rules cannot be too large — these requirements *implicitly* approximate Kolmogorov complexity

Section 2

What gives neural networks their power?

Structure of a neural network

- 1 neuron is a **dot product** followed by a **non-linearity**:

$$\sigma \langle \boldsymbol{x}, \boldsymbol{w} \rangle \tag{1}$$

- The non-linearity can take various forms, *eg*:

$$\sigma(\xi) = \frac{1}{1 + e^{-\xi}} \tag{2}$$

Structure of a neural network

- 1 **layer** of neurons is a matrix multiplication:

$$\sigma(\ W \cdot \boldsymbol{x}\ ) \tag{3}$$

- A neural **network** is the function **composition** $(f \circ f)$ of many layers:

$$[\sigma W]^L\ \boldsymbol{x} \tag{4}$$

Properties of neural networks

- An NN is a function with many **parameters**
- It is a **universal function approximator** [Cybenko 1989]
- Its proof can be traced to Weierstrauss's approximation theorem (1885): any continuous function can be uniformly approximated by polynomials
- But the proof is independent of depth

Power of NNs comes from depth

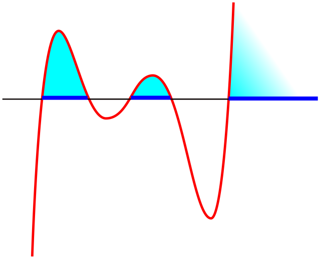
- Suppose $\sigma W \boldsymbol{x}$ is a *cubic* polynomial
- Adding each layer is equivalent to:

$$(\text{polynomial} \circ \text{polynomial}) \tag{5}$$

- Thus, the resulting polynomial has total degree $= 3^L$
- In other words, total degree grows **exponentially**

Power of NNs comes from depth

- **Fundamental theorem of algebra**: polynomial **degree** $=$ zero-crossings of curve with $x = 0$



(6)

- In higher dimensions: how many pieces does a surface carve up the space

Power of NNs comes from depth

- Same idea as **VC-dimension** [Vapnik–Chervonenkis 1971]
- VC-dim $=$ max $\#$ pieces the ambient space is *shattered* by a family of functions
- My conjecture: VC-dim of multi-layer NN grows exponentially as $\#$ layers
- Contradicts with current bound $= O(W \log W)$ where $W =$ total $\#$ parameters; I will investigate further

Power of NNs comes from depth

- The **exponential growth** of VC-dim means that NNs can represent highly complex **families** of functions
- and with a relatively small # of parameters, which can be implemented on a computer

Revelation from convolutional NNs

- Yann LeCun in 1989 invented **ConvNets**, which revolutionized machine vision, and got him a Turing Award

Revelation from convolutional NNs

- CNN replaces the conventional **dot product** with the **covolutional product**:

$$\boxed{\text{dot product}} \qquad \langle \boldsymbol{x}, \boldsymbol{w} \rangle \rightsquigarrow (f * g) \qquad \boxed{\text{convolution}} \tag{7}$$

- The convolution has **translation invariance**, which is suitable for vision:

$$T_x(f) * g = T_x(f * g) \tag{8}$$

- This is an **inductive bias**, that makes learning faster

Revelation from convolutional NNs

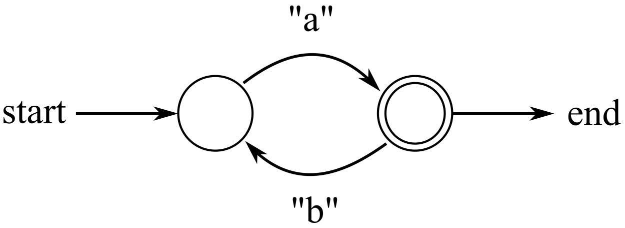
- Actually **vision** obeys **affine** invariance, *ie*: translations, rotations, dilations, ...
- It seems that merely translational invariance yields enough **acceleration** such that CNNs surpassed human vision performance in 2012
- Thus we see that **inductive bias** is still useful in deep learning

Section 3

Turing machines and universal logic

Finite state machines

- Finite state machines are usually defined by tuples (skipped here), *eg*:



(9)

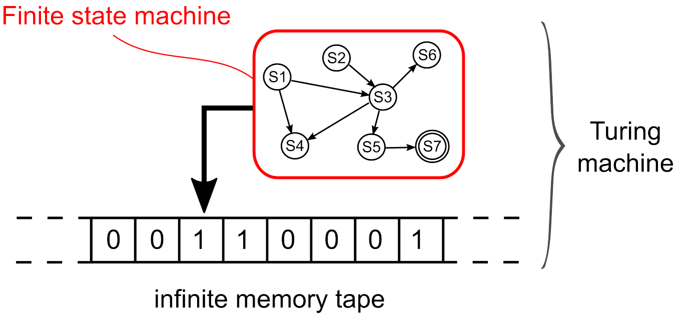
- This automata **accepts** strings like "a", "aba", "ababa..."

Finite state machines

- FSMs can accept strings of the form a^{m}b^n, where m and n are different
- But they can't recognize a^{k}b^k as they have no way to "remember" how many times k is
- The **languages** accepted by FSMs are called **regular languages**
- Formulated in 1950s by Noam Chomsky (computer scientist + linguist, now a leftist critic of US politics)

Turing machines

Turing machine = finite state machine + infinite **memory tape** (each state can read/write 1 symbol)



(10)

Turing machines

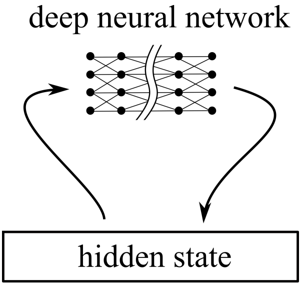
- Finite state machine $+$ infinite tape $=$ can compute all computable functions, *ie*, **Church-Turing hypothesis**
- Turing machines are **equivalent** to: λ-calculus, combinatory logic, cellular automata, game of life, recurrent neural networks, ... *etc*

Alan Turing (1912-1954)

- Turing was a man very much ahead of his time
- He considered neurons as learning machines
- Also considered evolutionary algorithms
- In 1940s there were no computers yet — he invented them!
- He formulated the form of **all computable functions**, thus **confining** the problem of AI within a framework

Recurrent neural networks

Structure of RNNs can be seen as similar to (10):



(11)

Key is that the **hidden state** can store **intermediate results** of computations, enabling RNNs to be Turing-universal

Section 4

Structure of classical AI systems

John McCarthy (1927-2011)

- "Father of AI"
- Held the first AI conference in 1956 in Dartmouth
- Pioneered the use of **mathematical logic** as **knowledge representation** in AI
- In later years he studied **term rewriting systems**, a more **generalized** form of logic

The world of logical structures

proof calculi
of various logics, including HOL

Curry-Howard isomorphism $\equiv$

Cartesian-closed categories

$\equiv$ equivalence of categories

Scott domains

models of

combinatory logic

$\equiv$

relation algebra

relation algebra is somewhat equivalent to predicate logic but in a non-obvious way

typed λ-calculus
(simple type theory)

$\subset$ more expressive / powerful

untyped λ-calculus

Church-Turing hypothesis says that λ-calculus captures universal computation

$\equiv$

predicate logic
(first-order logic)

$\subset$

higher-order logic

predicate logic is a "fibration" over the base category of propositional logic

more "traditional" mathematical structures below this line

propositional logic

probabilistic form $\sim$

Bayesian networks

modeled by

Boolean algebra $\cong$ **topologies**
(open sets)

Propositional vs predicate logic

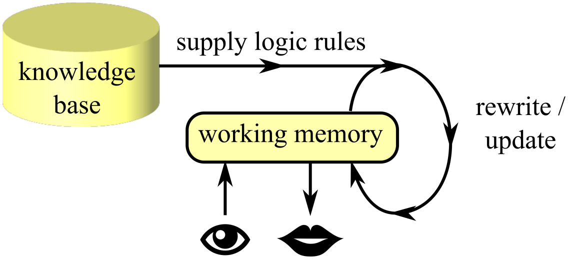
- The important distinction is between **propositional** and first-order **predicate** logic
- In propositional logic, propositions don't have **internal structure**

$$P_1 = \text{“It rained yesterday”}$$
$$P_2 = \text{“It's raining today”}$$

- Predicate logic: $P_3 = \text{rain(New York, today)}$
- Predicates bring about the complexity of **substitutions**

Architecture of logic-based AI systems

This cycle is as important as the *Carnot cycle* in the age of steam engines:



(12)

What is a logic rule?

- Example: loving someone and not loved back implies heartbreak:

$$\heartsuit(x, y) \wedge \neg\heartsuit(y, x) \Rightarrow \frownie(x) \tag{13}$$

- This is a rule. Variables x, y need to be **substituted** with appropriate objects, *eg* $\{x \setminus \text{John}, y \setminus \text{Mary}\}$
- Algorithm for finding substitutions is called **matching** or **unify**

SOAR architecture

- is a famous **cognitive architecture**
- SOAR searches for logic rules that match with working memory, similar to what is depicted in (12)
- SOAR uses the **Rete** algorithm to search for applicable rules efficiently, this is an important algorithm in classical AI (*rete* meaning "web-like" in Latin)

My theory

- In my theory, the rules and their matching mechanism are **absorbed** into an NN
- The NN is a weapon whose strength is in approximating very complex mappings; It is the most powerful machine-learning technology we have nowadays
- In my design I **relax** the structure of logic to get just the right amount of inductive bias
- In this design the central component is the **symmetric** NN

Commutativity of logic $\wedge$

- The commutativity of $\wedge$ is perhaps the most important law of logic, *eg*:

$$\text{hungry} \wedge \text{penniless} \Leftrightarrow \text{penniless} \wedge \text{hungry} \tag{14}$$

- Could be understood as: When deducing a conclusion from some premises, the propositions in the premise could be presented in **any order**, even containing irrelevant propositions
- The commutative law abstracts the structure of propositions, similar to the importance of commutative groups (also called Abelian group, in memory of Abel) in abstract algebra

Symmetric neural networks

- Convolutional NNs possess translational invariance
- Similarly, **symmetric** NNs possess **permutation invariance**
- This can be realized via **weight-sharing**, as similar in ConvNets
- SymNet : logic $\approx$ ConvNet : vision

Will China build its own AGI?

- The failure of Japan in 1980s to develop 5th-Generation computers can be a lesson
- As the earth has limited resources, technological progress often is the result of competition among national entities
- In the US there are people against China; In China there are also people dragging our feet, supporting outsiders. Yet I have also recieved help from American friends
- Therefore, I tend to be more supportive of globalized AGI projects

Thanks for watching ☺