TTIC 31230, Fundamentals of Deep Learning

David McAllester, April 2019

The Quest for Artificial General Intelligence (AGI)

Review: Chomsky vs. Kolmogorov

Noam Chomsky: By the no free lunch theorem **natural language grammar is unlearnable without an innate linguistic capacity**. In any domain a strong prior (a learning bias) is required.

Leonid Levin, Andrey Kolmogorov, Geoff Hinton and Jürgen Schmidhuber: **Universal learning algorithms exist. No domain-specific innate knowledge is required.**

Review: The Free Lunch Theorem

Consider any fixed language for naming functions. For example C++. (or English?)

Let $|f|$ be the number of bits it takes to name function f in that langauge.

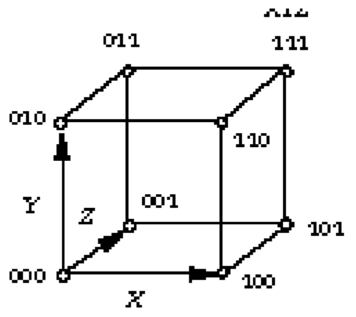
For $z \sim \mathrm{Pop}$, let $\mathcal{L}(f, z) \in [0, L_{\max}]$ be a bounded loss function.

Free Lunch Theorem: With probability at least $1-\delta$ over the draw of training data $z_1, \ldots, z_N$ from Pop, the following holds *simultaneously* for all namable functions f

$$\mathcal{L}(f) \leq \frac{10}{9}\left(\widehat{\mathcal{L}}(f) + \frac{5L_{\max}}{N}\left((\ln 2)|f| + \ln\frac{1}{\delta}\right)\right)$$

Function Representations

Consider continuous functions $f : [0,1]^N \to \mathbb{R}$



$$\xrightarrow{f} \quad \mathbb{R}$$

Given the corner values, the interior can be filled.

$$f(x_1, \ldots, x_N) = E_{y_1,\ldots,y_N \sim \text{Round}(x_1,\ldots,x_N)}\, f(y_1, \ldots, y_n)$$

Hence each of the 2^N corners has an independent value.

The Kolmogorov-Arnold representation theorem (1956)

For continuous $f : [0,1]^N \to \mathbb{R}$ there exists continuous "activation functions" $\sigma_i : \mathbb{R} \to \mathbb{R}$ and continuous $w_{i,j} : \mathbb{R} \to \mathbb{R}$ such that

$$f(x_1, \ldots, x_N) = \sum_{i=1}^{2N+1} \sigma_i \left(\sum_{j=1}^{N} w_{i,j}(x_j) \right)$$

A Simpler, Similar Theorem

For (possibly discontinuous) $f : [0,1]^N \to \mathbb{R}$ there exists (possibly discontinuous) $\sigma, w_i : \mathbb{R} \to \mathbb{R}$.

$$f(x_1,\ \ldots,\ x_N) = \sigma\left(\sum_i w_i(x_i)\right)$$

Proof: Select w_i to spread out the digits of its argument so that $\sum_i w_i(x_i)$ contains all the digits of all the x_i.

Cybenko's Universal Approximation Theorem (1989)

For continuous $f : [0,1]^N \to \mathbb{R}$ and $\varepsilon > 0$ there exists

$$\begin{aligned} F(x) &= \alpha^\top \sigma(Wx + \beta) \\ &= \sum_i \alpha_i \sigma\left(\sum_j W_{i,j}\, x_j + \beta_i\right) \end{aligned}$$

such that for all x in $[0,1]^N$ we have $|F(x) - f(x)| < \varepsilon$.

How Many Hidden Units?

Consider Boolean functions $f: \{0,1\}^N \to \{0,1\}$.

For Boolean functions we can simply list the inputs $x^0, \ldots, x^k$ where the function is true.

$$f(x) = \sum_k \mathbf{1}[x = x^k]$$

$$\mathbf{1}[x = x^k] \approx \sigma\left(\sum_i W_{k,i} x_i + b_k\right)$$

A simpler statement is that any Boolean function can be put in disjunctive normal form.

Representing Functions as IO Tables

Both of the previous theorems implicitly treat functions as tables of intput-output pairs.

☹️

Representing Functions by Circuits

We can define functions of Boolean variables (the corners of $[0,1]^d$) with Boolean circuits or linear threshold circuits.

We can define functions of $[0,1]^d$ with feed-forward real-valued networks (Deep models).

Circuit Complexity Theory

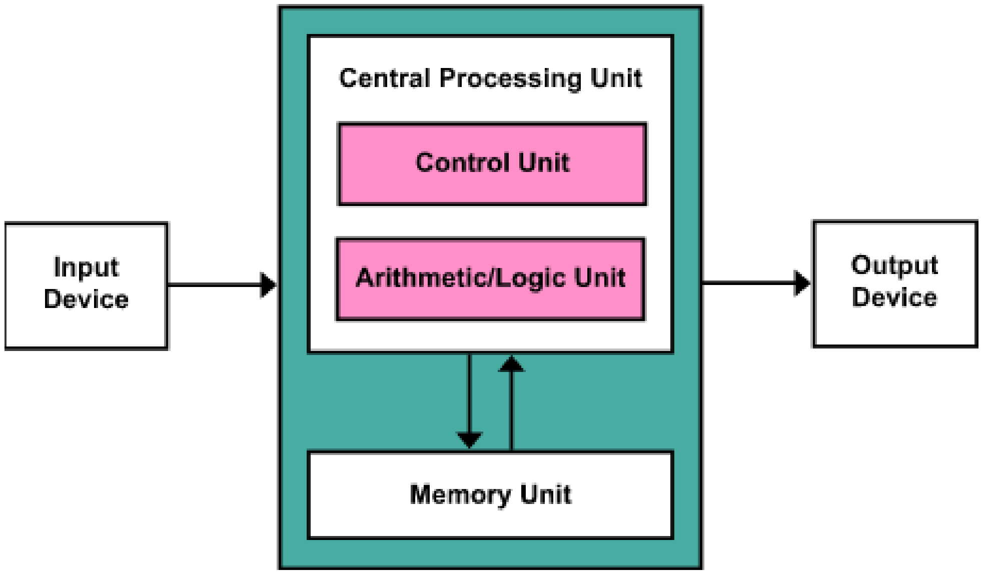
Building on work of Ajtai, Sipser and others, Hastad proved (1987) that any bounded-depth Boolean circuit computing the parity function must have exponential size.

Matus Telgarsky recently gave some formal conditions under which shallow networks provably require exponentially more parameters than deeper networks (COLT 2016).

Representing Functions with Programs

Neural Turing Machines Alex Graves, Greg Wayne, Ivo Danihelka, 2014

(Actually a differentiable Von Neumann architecture)



The machine undergoes discrete time state transitions defined a differentiable feed-forward circuit.

Discrete Time Differentiable State Transitions

The state is defined by a state vector — perhaps analogous to the hidden state of a gated RNN — plus read address registers and write address registers and a memory consisting of a (larger) set of data registers.

Reading and writing to the data registers involves attentions over the memory defined by the address registers.

An Execution Cycle

The CPU receives an external input.

The first step is to recompute the attention vectors in the read address registers.

The CPU computes a “key” k^h for each head h and an attention α^h_j

$$\begin{aligned} \alpha^h_j &= \operatorname*{softmax}_j \; k^h \cdot M[j] \\ r^h &= \sum_j \alpha^h_j M[j] \end{aligned}$$

Reading from Memory

Once the read address vectors (attentions) have been computed we read a value for each read head.

$$v^h = \sum_j \alpha_j^h M[j]$$

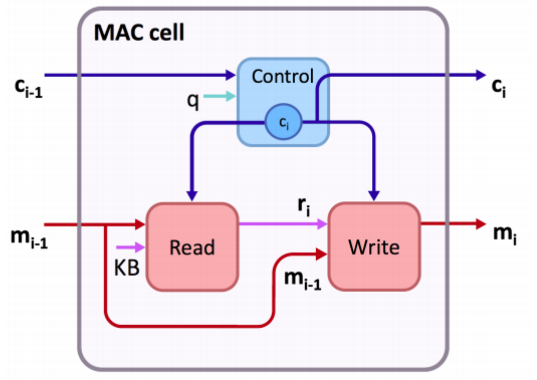
After reading from memory the machine computes a key and attention for each write head.

Each write operation involves “forget” and “input” operation analogous to an LSTM.

Finally the next hidden state is computed and emitted.

Compositional Attention Networks for Machine Reasoning

Hudson and Manning, ICLR 2018



The MAC cell is similar to a gated RNN cell used as the decoder in translation.

It is also similar to a Neural Turing Machine.

It was applied to image-based question answering and uses attention over the image and the question during multi-step "decoding".

The Turing Tarpit

The choice of programming language does not matter.

For any two Turing universal languages L_1 and L_2 there exists a compiler $C : L_1 \to L_2$ such that

$$|C(h)| \leq |h| + |C|$$

Deep Learning for NLP vs. NLP for Deep Learning

Progress in an application area, such as vision, can lead to progress in deep learning, such as ResNet.

What can we learn about learning and representation (AGI) by working in NLP?

Will the Transformer (motivated by phrase structure) replace ResNet?

The Procedural vs. Declarative Debate

Any Discussion today of the "knowledge representation problem" is likely to entail a debate between proponents of **declarative** and **procedural** representations of knowledge.

Terry Winograd, 1974

Programming Languages (Procedural)

vs. Natural Language (Declarative)

Natural Language Semantics

Thousands of civilians have fled advances by Syrian government forces in eastern Ghouta as ...

Stanford Parse Tree

```
(NP (NP (NNS Thousands))
    (PP (IN of) (NP (NNS civilians))))
(VP (VBP have)
    (VP (VBN fled)
        (NP (NNS advances))
        (PP (IN by) (NP (NP (JJ Syrian)
                            (NN government)
                            (NNS forces))
                        (PP (IN in) (NP (JJ eastern)
                                        (NNP Ghouta))))))
```

Stanford Dependencies

```
root(ROOT-0, fled-5)
  aux(fled-5, have-4)
  nsubj(fled-5, Thousands-1)
    nmod(Thousands-1, civilians-3)
      case(civilians-3, of-2)
  dobj(fled-5, advances-6)
  nmod(fled-5, forces-10)
    case(forces-10, by-7)
    amod(forces-10, Syrian-8)
    compound(forces-10, government-9)
    nmod(forces-10, Ghouta-13)
      case(Ghouta-13, in-11)
      amod(Ghouta-13, eastern-12)
```

Just Parantheses

```
(Thousands of civilians)
(have fled)
(advances (by (Syrian government forces))
          (in eastern Ghouta))
```

Natural Language: Parsing

A Fast and Accurate Dependency Parser using Neural Networks, Danqi Chen and Christopher Manning, 2014.

$$\begin{array}{l} \vdots \\ s_3 \\ s_2 \\ s_1 \\ * \quad b_1\ b_2\ b_3\ \ldots \end{array} \quad \overset{\text{push}}{\Rightarrow} \quad \begin{array}{l} \vdots \\ s_2 \\ s_1 \\ b_1 \\ * \quad b_2\ b_3\ b_4\ \ldots \end{array}$$

Arc Transitions

$$\begin{array}{l} \vdots \\ s_3 \\ s_2 \\ s_1 \\ * \quad b_1\ b_2\ b_3\ \ldots \end{array} \quad \overset{\text{LeftArc}}{\Longrightarrow} \quad \begin{array}{l} \vdots \\ s_4 \\ s_3 \\ s_1 \\ * \quad b_1\ b_2\ b_3\ \ldots \end{array} \qquad \text{Emits } s_1 \xrightarrow{L} s_2$$

$$\begin{array}{l} \vdots \\ s_3 \\ s_2 \\ s_1 \\ * \quad b_1\ b_2\ b_3\ \ldots \end{array} \quad \overset{\text{LeftArc}}{\Longrightarrow} \quad \begin{array}{l} \vdots \\ s_4 \\ s_3 \\ s_2 \\ * \quad b_1\ b_2\ b_3\ \ldots \end{array} \qquad \text{Emits } s_2 \xrightarrow{R} s_1$$

Dependency Parsing Machine Configurations

A machine configuration

$$c = (s, b, A)$$

$$\begin{aligned} s &\sim \text{stack} \\ b &\sim \text{buffer} \\ A &\sim \text{Dependency Arcs} \end{aligned}$$

Training

Construct a database of machine configurations labeled with actions.

Train an MLP with one hidden layer to a softmax over actions and train on cross entropy.

The input to the MLP is a concatenation of 18 word vectors defined in terms of the configuration

plus 18 corresponding part of speech vectors

and 12 parent edge label vectors.

Reference (Entity Linking)

Thousands of civilians have fled advances by Syrian government forces in eastern Ghouta as Damascus makes rapid gains against the last major rebel enclave near the capital.

Damascus $\Rightarrow$ Assad

Rapid Gains $\Rightarrow$ advances-6

the last major rebel enclave ... $\Rightarrow$ Ghouta

the capital $\Rightarrow$ Damascus

Reference vs. Composition

Functional programming is compositional

$$x = f(y, z)$$

The meaning of x is computed by f from the meaning of y and z.

But in language we typically have that $f(y, z)$ is a mention and x is its referent.

(the last (major rebel enclave) (near (the capital)))

$$x = (\text{the last } Q\ P)$$

Logical Representations of Events

Let e range over "events".

$$e : \mathrm{give}(a_1, x, a_2) \Rightarrow \mathrm{had}(a_1, x, \mathrm{before}(e)) \wedge \mathrm{had}(a_2, x, \mathrm{after}(e))$$

This is related to Davidsonian semantics for natural language (1969) and the situation calculus of McCarthy and Hayes (1968).

Bottom-up Logic Programming

Bottom-up logic programming is distinguished by its relationship to dynamic programming algorithms.

$$\mathrm{At}(x) \Rightarrow \mathrm{Reachable}(x)$$

$$\mathrm{Reachable}(x) \wedge \mathrm{CanGo}(x, y) \Rightarrow \mathrm{Reachable}(y)$$

This defines a linear time algorithm for reachability.

Datalog

A set of inference rules each of which has antecedents and conclusions that are just predicates applied to variables is called a **datalog** program.

It can be shown that datalog "captures the complexity class P" — they can express **all and only** polynomial time decidable relations (provided the entities are assigned a total order).

General bottom-up logic programs, including expressions (terms), are Turing complete.

$$N(x) \Rightarrow N(s(x))$$

The Curry-Howard Isomorphism

This is an isomorphism between “proofs” and “programs”.

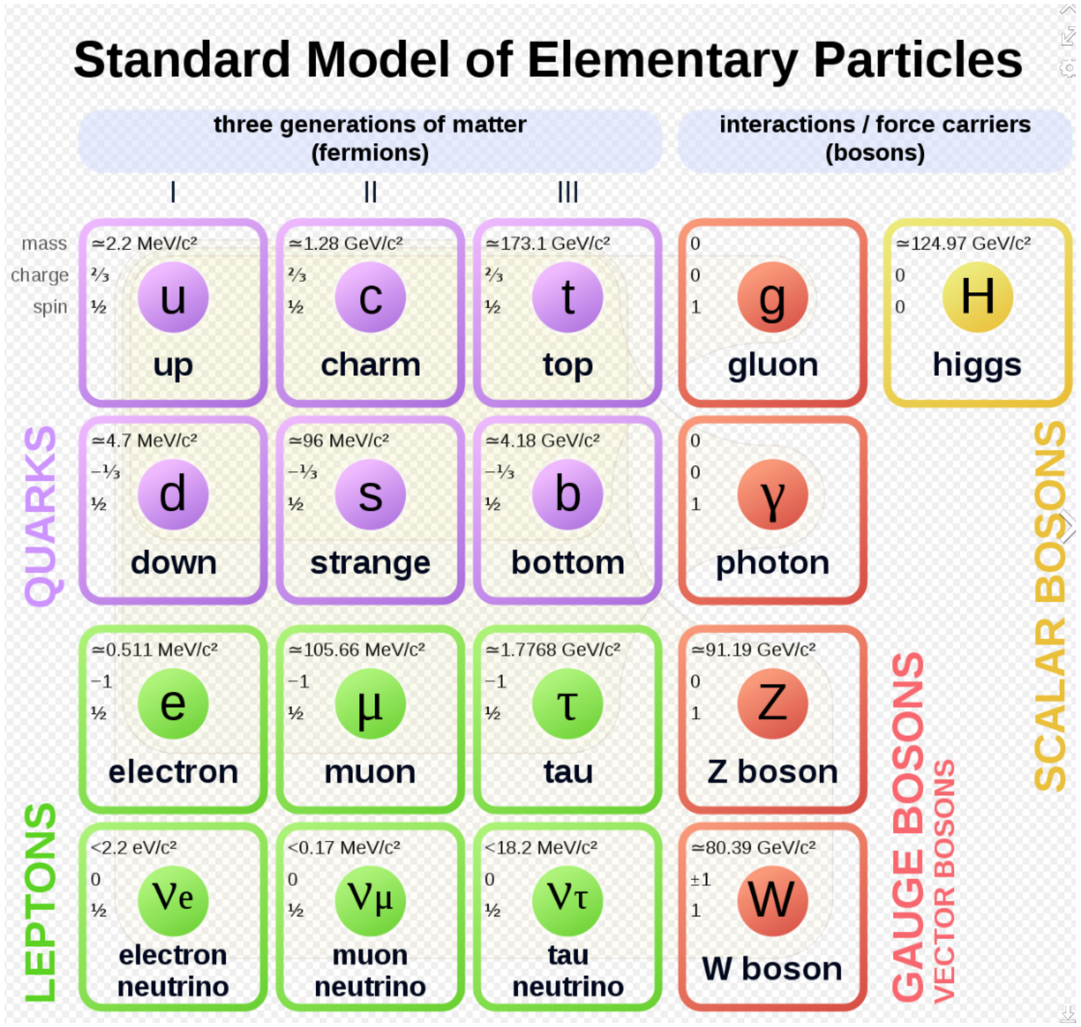
A proof that x is reachable is a path to x — a path that can actually be taken to get there.

Curry-Howard: $(\exists f : \sigma \to \tau) \quad \Leftrightarrow \quad (\exists \text{Proof} : (\exists \sigma) \Rightarrow (\exists \tau))$.

But this is not really true — while constructive proofs are useful, we also believe in non-constructive proofs.

Furthermore, “proof irrelevance” is essential to dynamic programming.

The 17 Fundamental Particles



The 14 Fundamental Constructs of Mathematics

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

The Substitution of Isomorphics

The isomorphism relation $u =_\sigma v$ is challenging to define in complete generality.

But we know we want the following substitution rule.

$$\begin{array}{l} \Sigma \vdash \sigma : \mathbf{Type} \\ \Sigma;\ x:\tau \vdash e[x]:\sigma \\ \Sigma \vdash a =_\tau\ b \\ \hline \Sigma \vdash e[a] =_\sigma e[b] \end{array}$$

The Baldwin Effect: Learning Facilitates Adaptation

In a 1987 paper entitled “How Learning Can Guide Evolution”, Goeffrey Hinton and Steve Nowlan brought attention to a paper by Baldwin (1896).

The basic idea is that by facilitating adaptation of the individual, learning facilitates evolution of the species.

For example, longer arms are easier to evolve if arm control is learned — arm control is then independent of arm length. Arm control and arm structure become more modular.

The Meta-Baldwin Effect: Learning Facilitates Learning

The Baldwin effect should apply to brain modules as well, such as vision or the motor cortex.

By facilitating adaptation of the use of the vision module, learning facilitates the evolution of the vision module.

By facilitating adaptation of the use of the vision module, use-learning facilitates internal-learning in the vision module.

The Universality Assumption in Learning and Levin's Universal Problem Solver

Leonid Levin observed that one can construct a universal solver. The solver takes as input a solution tester and returns as output a solution whenever a solution exists.

Levin's solver is universal in the sense that it is not more than a constant factor slower than any other solver mapping testers to solutions.

It follows that for any problem in NP, the universal solver provides a P-time algorithm whenever any such algorithm exists.

Levin's Universal Solver

We time share all programs giving time slice $2^{-|h|}$ to program h where $|h|$ is the length in bits of h.

The run time of the universal solver is at most

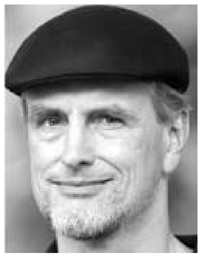
$$O(2^{-|h|}(h(n) + T(n)))$$

where $h(n)$ is the time required to run program h on a problem of size n and $T(n)$ is the time required to check the solution.

Here $2^{-|h|}$ is independent of n and is technically a constant.

Logic vs. Learning

Logical theorem proving was viewed in 1960s as the path to general problem solving. (The "general problem solver" (GPS) Simon, Shaw and Newel, 1959).



Schmidhuber's answer to the failure of logic is to assume universality of a learning algorithm for learning a general problem solver. The Optimally Ordered Problem Solver (OOPS), Schmidhuber, 2002.

An Intelligence Chain Reaction?

Let an ultraintelligent machine be defined as a machine that can far surpass all the intellectual activities of any person however clever. Since the design of machines is one of these intellectual activities, an ultraintelligent machine could design even better machines; there would then unquestionably be an intelligence explosion, and the intelligence of humanity would be left far behind. Thus the first ultraintelligent machine is the last invention that humanity need ever make, provided that the machine is docile enough to tell us how to keep it under control.

I.J. Good, 1969

END