TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2020

Representing Functions with Programs

Python, Assembler, and the Turing Tarpit

Chomsky vs. Kolmogorov and Hinton

Noam Chomsky: Natural language grammar cannot be learned by a universal learning algorithm. This position is supported by the "no free lunch theorem".

Andrey Kolmogorov, Geoff Hinton: Universal learning algorithms exist. This position is supported by the "free lunch theorem".

The No Free Lunch Theorem

Without prior knowledge, such as universal grammar, it is impossible to make a prediction for an input you have not seen in the training data.

Proof: Select a predictor h uniformly at random from all functions from $\mathcal{X}$ to $\mathcal{Y}$ and then take the data distribution to draw pairs $(x, h(x))$ where x is drawn uniformly from $\mathcal{X}$. No learning algorithm can predict $h(x)$ where x does not occur in the training data.

The Occam Guarantee (Free Lunch Theorem)

Consider a classifier f written in C++ with an arbitrarily large standard library.

Let $|f|$ be the number of bits needed to represent f.

The Occam Guarantee (Free Lunch Theorem)

$$0 \leq \mathcal{L}(h, x, y) \leq L_{\mathrm{max}}$$

$$\begin{aligned}\mathcal{L}(h) &= E_{(x,y)\sim \mathrm{Pop}}\, \mathcal{L}(h, x, y) \\ \hat{\mathcal{L}}(h) &= E_{(x,y)\sim \mathrm{Train}}\, \mathcal{L}(h, x, y)\end{aligned}$$

Theorem: With probability at least $1 - \delta$ over the draw of the training data the following holds simultaneously for all f.

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

Representing Functions with Programs

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

(Actually a differentiable Von Neumann architecture)

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

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".

What about Python?

High level scripting languages such as Python seem to be the most productive programming languages for human programmers.

Does Python represent a particularly efective universal learning bias?

Productivity in programming seems to be greatly enhanced by functional expressions (functional programming) and object-oriented programming (objects, classes an inheretitance).

This seems crucial if we want to somehow achieve I. J. Good's intelligence explosion.

The Turing Tarpit

But in theory the choice of programming language does not matter.

For any two Turing universal languages, say Python and Assmbler, there exists an interpreter I for Python written in Assembler where write $I(h)$ for the assember interpreter I applied to Python program h. We then get

$$|I(h)|_{\text{Assembler}} = |h|_{\text{Python}} + |I|_{\text{Assembler}}$$

Bootstrapping layers of language can make the interpreter small.

The Turing Tarpit

$$|I(h)|_{\text{Assembler}} = |h|_{\text{Python}} + |I|_{\text{Assembler}}$$

Up to the additive constant of the interpreter, assembler gives just as good a learning bias as Python.

Yet we know that the choice of language does matter — Python is clearly better than assembler.

END