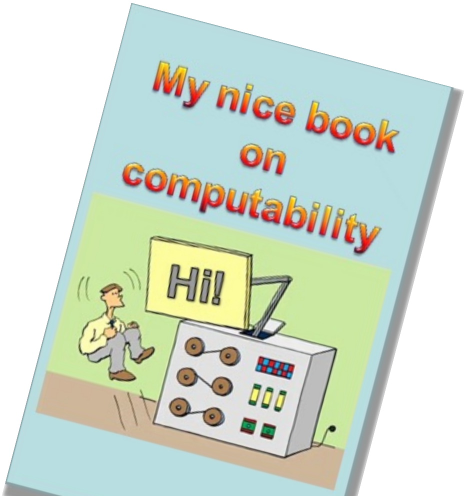
Are computability questions still relevant today?

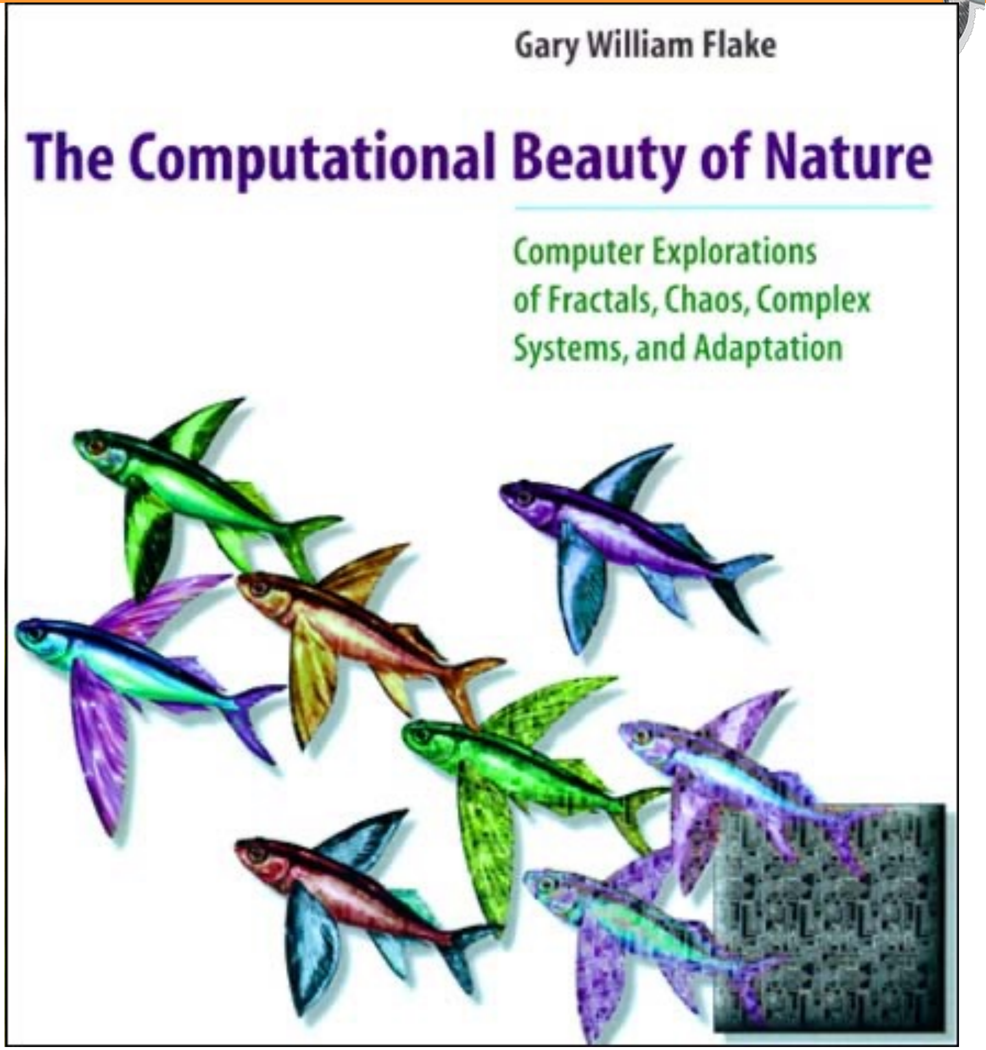
Gerard Vreeswijk,

cs-NLP seminar, Nov. ‘23



Why the flying fish?

- BSc level 2 course “Inl. Adaptieve Systemen” (2000-now)
- IAS was 2000-2008 taught by Marco Wiering, who knew a lot about reinforcement learning
- IAS will in 2025 likely be replaced by a course entitled “Nature-Inspired Computing” (Dutch title t.b.a.)



Gary William Flake

The Computational Beauty of Nature

Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation

Short bio

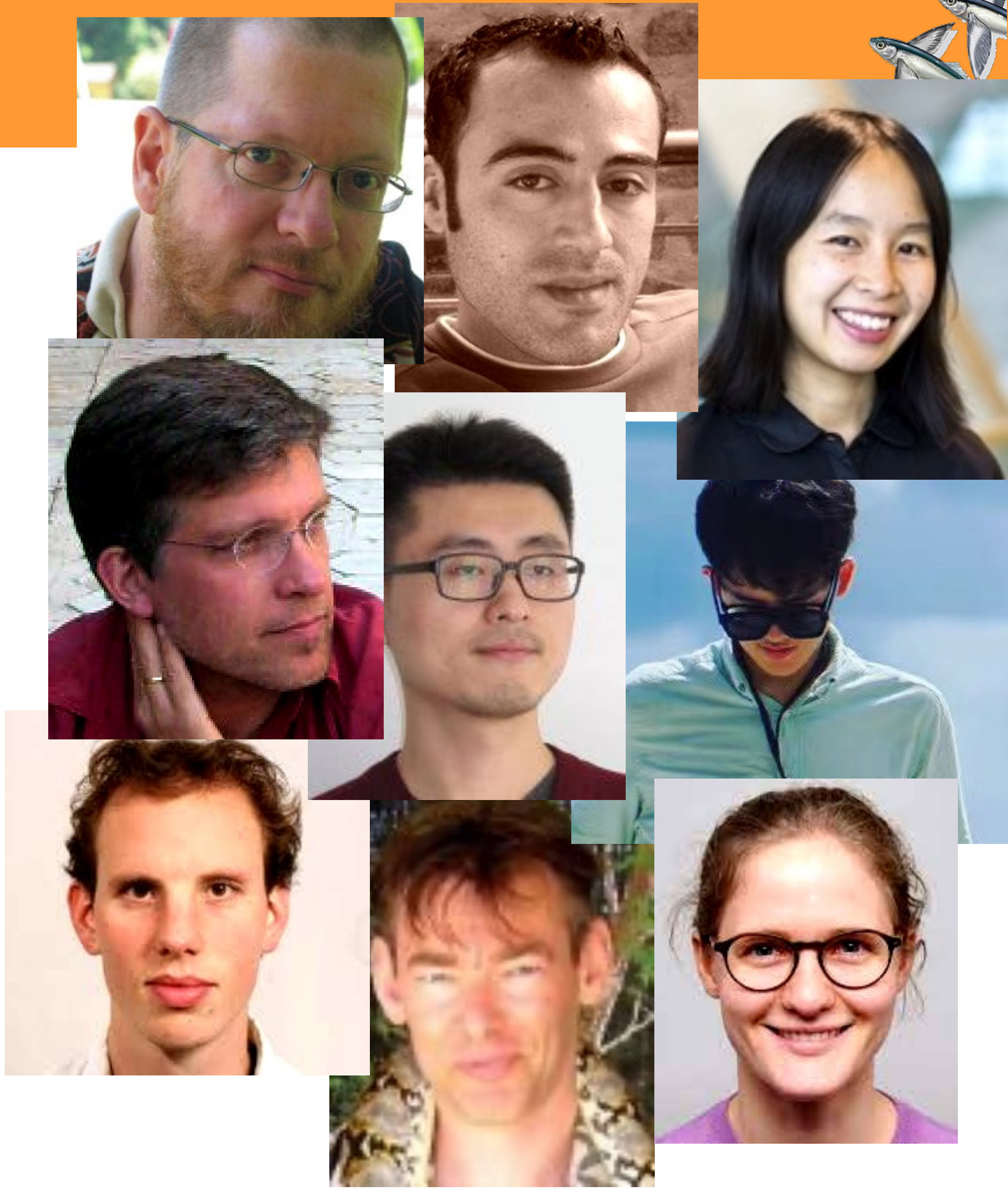
- Studied pure mathematics in Amsterdam (1986-1990)
- PhD on a KR&R topic (1993)
- Went into industry (ING insurances)
- Returned to academia (1995: Maastricht, 1997: Groningen, 1999: Utrecht)

  Role / agenda: ?
- Joined NLP group (2019)

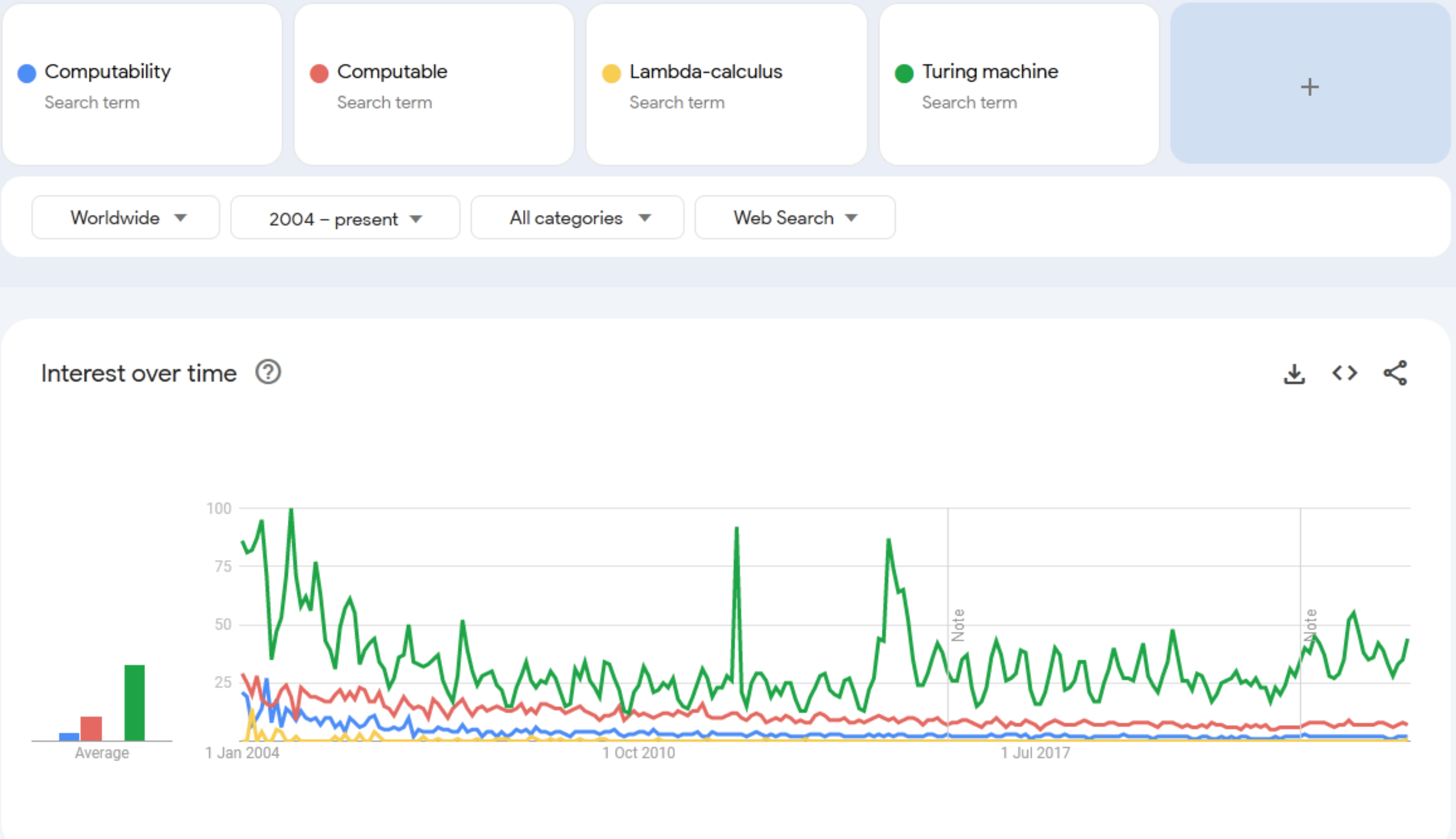
31 years (couldn't find another photo of myself in office)

My connection with NLP@cs

- Around 2018, the Intelligent Systems group changed chairs.
- No longer contribute to symbolic AI a.k.a. knowledge representation and reasoning.
- NLP is where the action is ...



Is computability still relevant today? Ask Google trends



● Computability
Search term

● Computable
Search term

● Lambda-calculus
Search term

● Turing machine
Search term

+

Worldwide ▾

2004 – present ▾

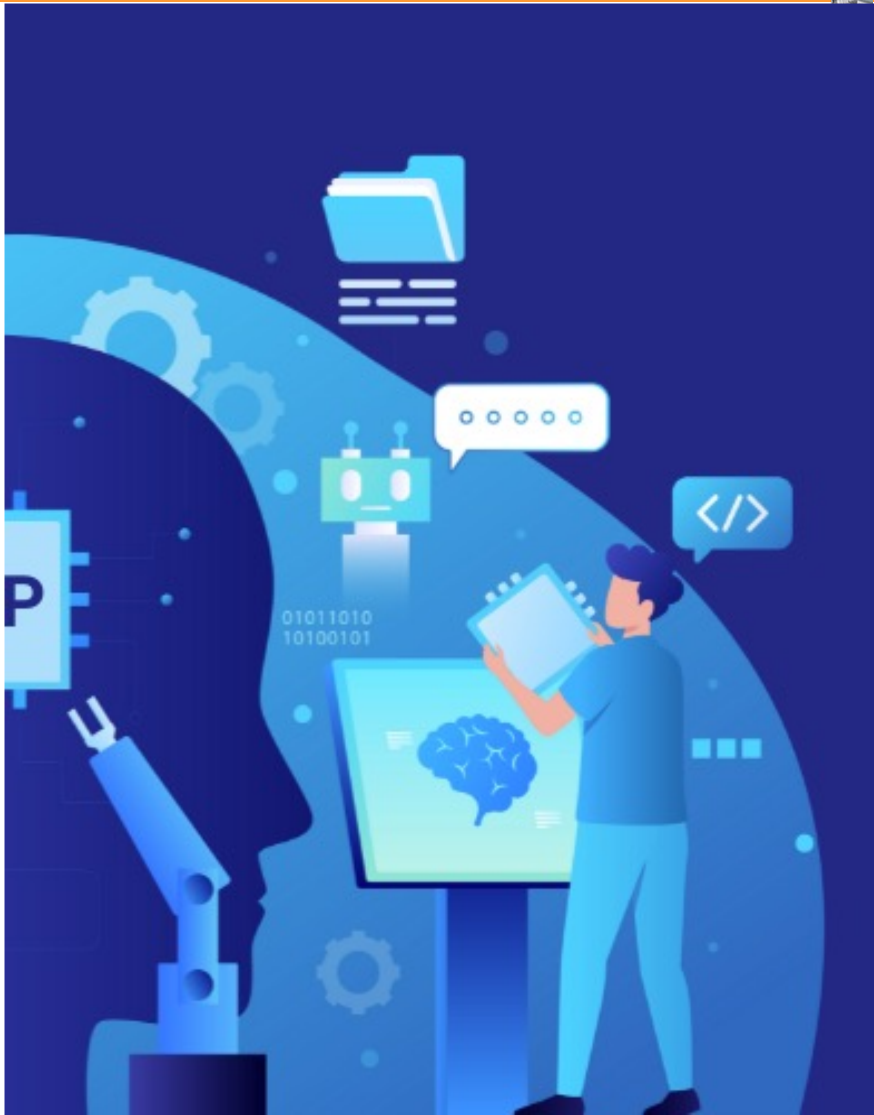
All categories ▾

Web Search ▾

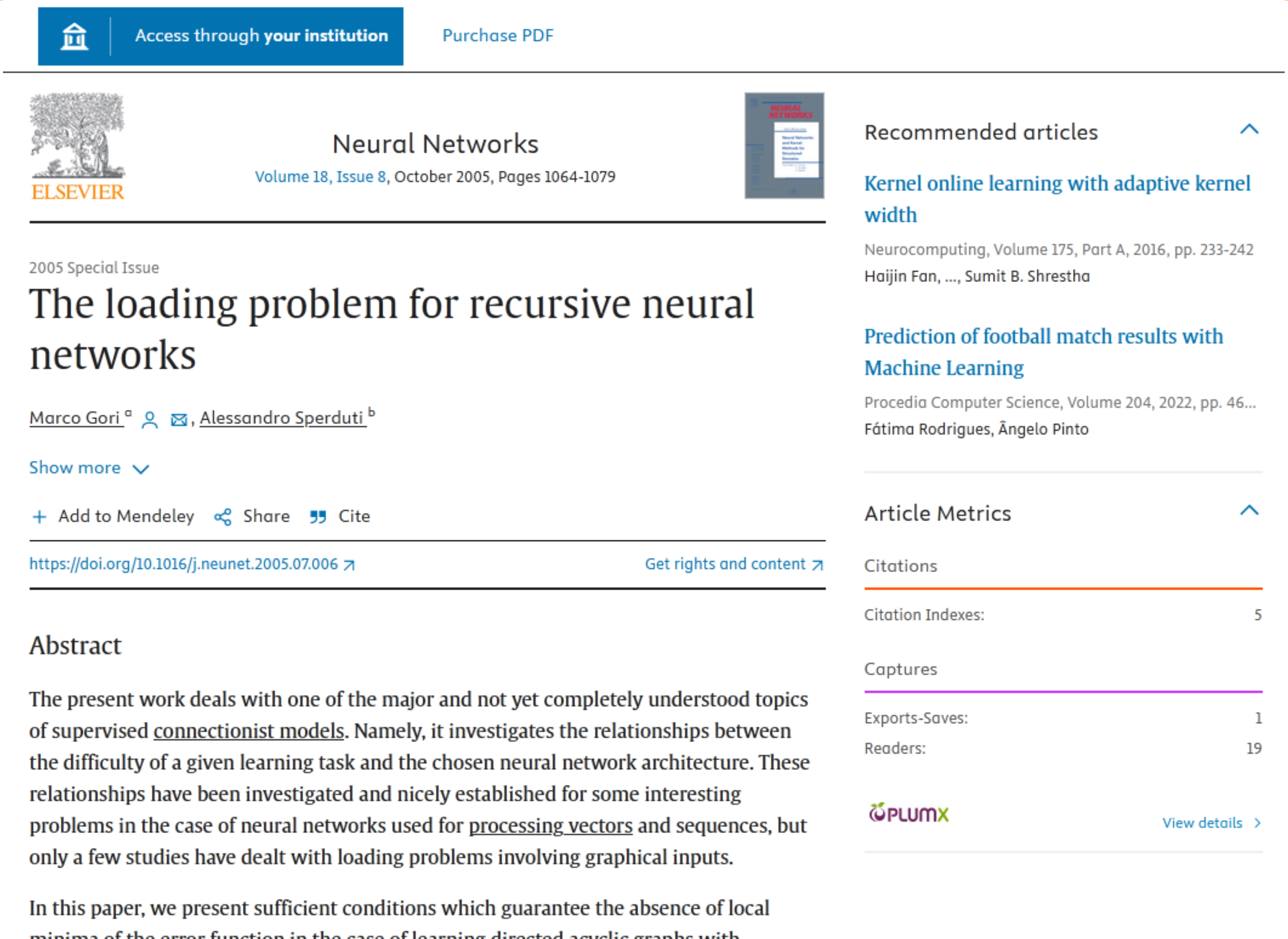
Interest over time ⓘ

Relation computability ↔ NLP

- Post’s correspondence problem
- Kolmogorov complexity
- The equivalence of context-free grammars
- Training of neural networks: the “loading problem” (Wiklicky, 1994; Gori *et al*., 2005)
- Underfitting in neural networks (Sehra *et al*., 2021)



The loading problem (2005)



ELSEVIER

Neural Networks
Volume 18, Issue 8, October 2005, Pages 1064-1079

2005 Special Issue

The loading problem for recursive neural networks

Marco Gori [a], Alessandro Sperduti [b]

https://doi.org/10.1016/j.neunet.2005.07.006

Abstract

The present work deals with one of the major and not yet completely understood topics of supervised connectionist models. Namely, it investigates the relationships between the difficulty of a given learning task and the chosen neural network architecture. These relationships have been investigated and nicely established for some interesting problems in the case of neural networks used for processing vectors and sequences, but only a few studies have dealt with loading problems involving graphical inputs.

In this paper, we present sufficient conditions which guarantee the absence of local minima of the error function in the case of learning directed acyclic graphs with

Recommended articles

Kernel online learning with adaptive kernel width

Neurocomputing, Volume 175, Part A, 2016, pp. 233-242
Haijin Fan, ..., Sumit B. Shrestha

Prediction of football match results with Machine Learning

Procedia Computer Science, Volume 204, 2022, pp. 46...
Fátima Rodrigues, Ângelo Pinto

Article Metrics

Citations

Citation Indexes:	5

Captures

Exports-Saves:	1
Readers:	19

The loading problem

Question: does there exists a training algorithm **A** that,

- for any learning task **T**, and
- backpropagation network **N**,

decides whether there exists a configuration of weights of **N** satisfying **T**?

Answer: no. Such an algorithm **A** does not exist, and never will.

Outline of a proof: reduce Hilbert's 10th problem to this one. [Hilbert's 10th problem is a famous question that was proven undecidable in 1970 by Y. Matiyasevich, *et al.*] □

A simple proof of Turing’s halting problem (1936)

The halting problem is undecidable (Turing)

Officially: Turing-complete

Theorem (Turing, 1936). Let J be a programming language. If J is sufficiently expressive, there cannot exist a program $h \in J$ that for every program/input combination $(j, i) \in J \times I$ can decide whether j with input i stops.

- Note: h is 2-placed and the programs j are 1-placed.

Proof (by contradiction):

- Suppose h does exist. So: $h(j, i) = 1 \Leftrightarrow j(i)$ stops.

Table for *h/2*

h	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	…
j_1	1	1	1	1	0	1	1	0	1	…
j_2	1	1	1	0	0	1	0	1	0	…
j_3	1	0	0	1	1	1	1	1	1	…
j_4	1	1	1	1	1	1	1	1	1	…
			1	0	0	1	1	0	1	…
			0	1	1	1	1	1	1	
j_8	0	1								

- Function *q* is different from all functions $j_i/1$.
- Therefore, *q* is not programmable.

- Using *h/2* it is possible to program function *q*:

  q(*i*): if *h(i, i) = 1* then loop forever else halt

- **Contradiction**.

  Our assumption is false: the function *h/2* cannot be programmed.

The perfect virus scanner does not exist

Theorem. Let *J* be a programming language. If *J* is sufficiently expressive, there cannot exist a program $v/1 \in J$ that determines for each file $i \in I$ whether *i* is a virus.

- **Proof**. Instances *j/0* of the inputless halting problem can be translated uniformly and automatically into a new program, call this *k/0* :

  run j/0 in quarantine; infect

- Now: *j/0* stops ⇔ *k/0* infects your PC.
- If *v/1* existed, then *v/1* could be used to determine whether *k/0* infects your PC, hence whether *j/0* halts.

Rice’s theorem

Theorem (Rice, 1957). Let J be a programming language, and let P be a (non-trivial) semantic (i.e., behavioral) property of programs. If J is sufficiently expressive, there cannot exist a program $g/1 \in J$ that determines for each program $j/0 \in J$ whether j satisfies P.

Examples of P:

- j prints a character
- j contains dead code
- j is functionally equivalent to some fixed program $k/0$

Proof. Similar to the non-existence proof of the universal virus scanner.

But now abstracts from the (code of the) virus scanner.

Books I consult the most

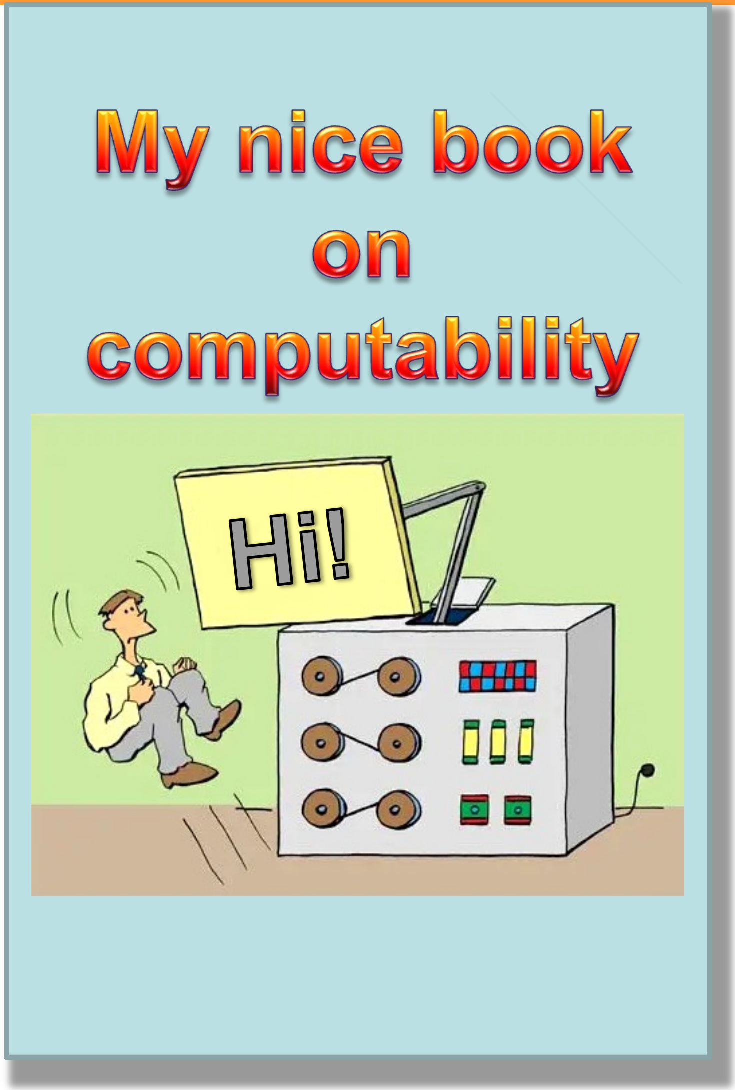
- **Authority and reference**: Handbook, Rogers (cyan), Odifreddi (yellow), ...
- **Intermediate**: Soare (orange)
- **“Introductory”**: Cooper (blue), Shen & Vereshchagin (brown), Cutland (pink), Weber (grey).

Problems with current books

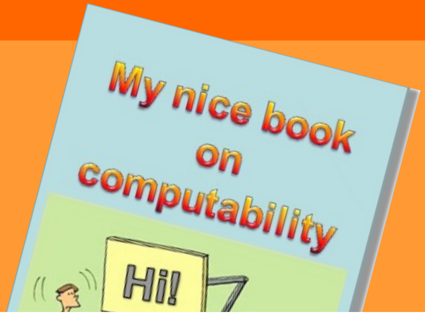
- Sub-optimal composition / organisation
- Jargon
- Exercises without answers
- To "deep"
- Too little context

Obstacles and pitfalls

- Focus on audience
- My constraints (e.g., knowledge of the field)
- Organisation
- Publisher
- Sample chapters (2)
- English
- Deadline



Current TOC



- Preliminaries
- Computable functions
- Computable approximations
- Decidable sets
- Semi-decidable sets
- Enumerable sets
- Creative sets (i.e., “natural” undecidable sets that are equivalent to the halting problem)
- The sets K and K_0
- Komogorov complexity
- Hypercomputation (interaction, true asynchronicity, updates)
- Randomness
- Models of computation, plus proofs that Turing machine ~ Markov algorithm ~ any higher programming language

Thank you 🙏