COLLECTIONELSS MACHINE LEARNING
THE OTHER SIDE OF AI

Marco Gori
Università di Siena

> Far better an approximate answer to the right question, which is often vague,
> than the exact answer to the wrong question, which can always be made precise.
> – John Tukey

S3P-2024

OUTLINE

Part I

1. The path of Artificial Intelligence
2. A paradigm-shift: The connectionist wave
3. Collectionless AI
4. Intelligence and laws of Nature

Part II

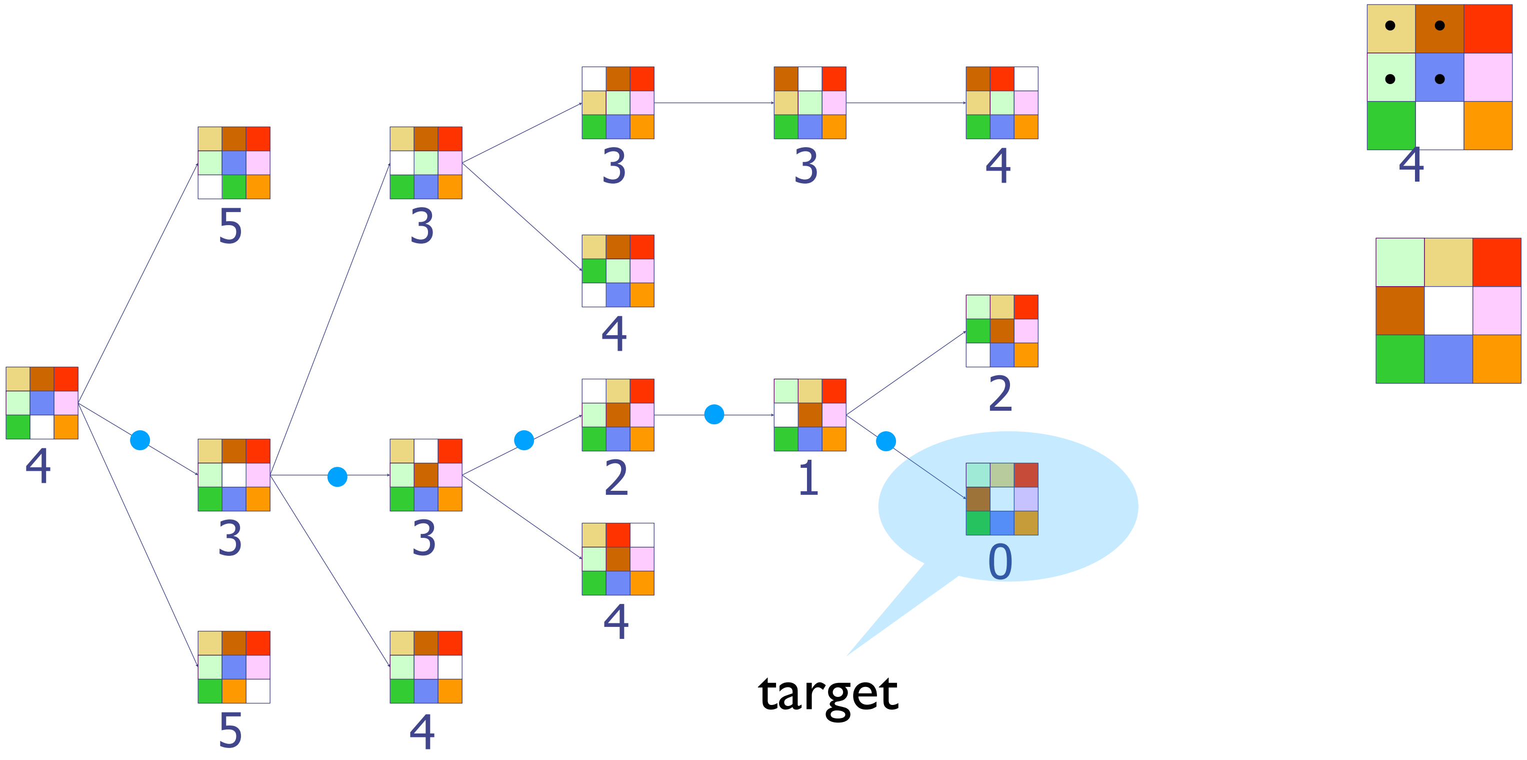
5. The Hamiltonian framework of learning
6. Cognidynamics: A Theory of Neural propagation
7. Neural vs wave propagation
8. Signal generation

THE PATH OF ARTIFICIAL INTELLIGENCE

my own view

WHERE THE QUALITY OF "ARTIFICIAL INTELLIGENCE" CAN EXCEED HUMAN INTELLIGENCE!

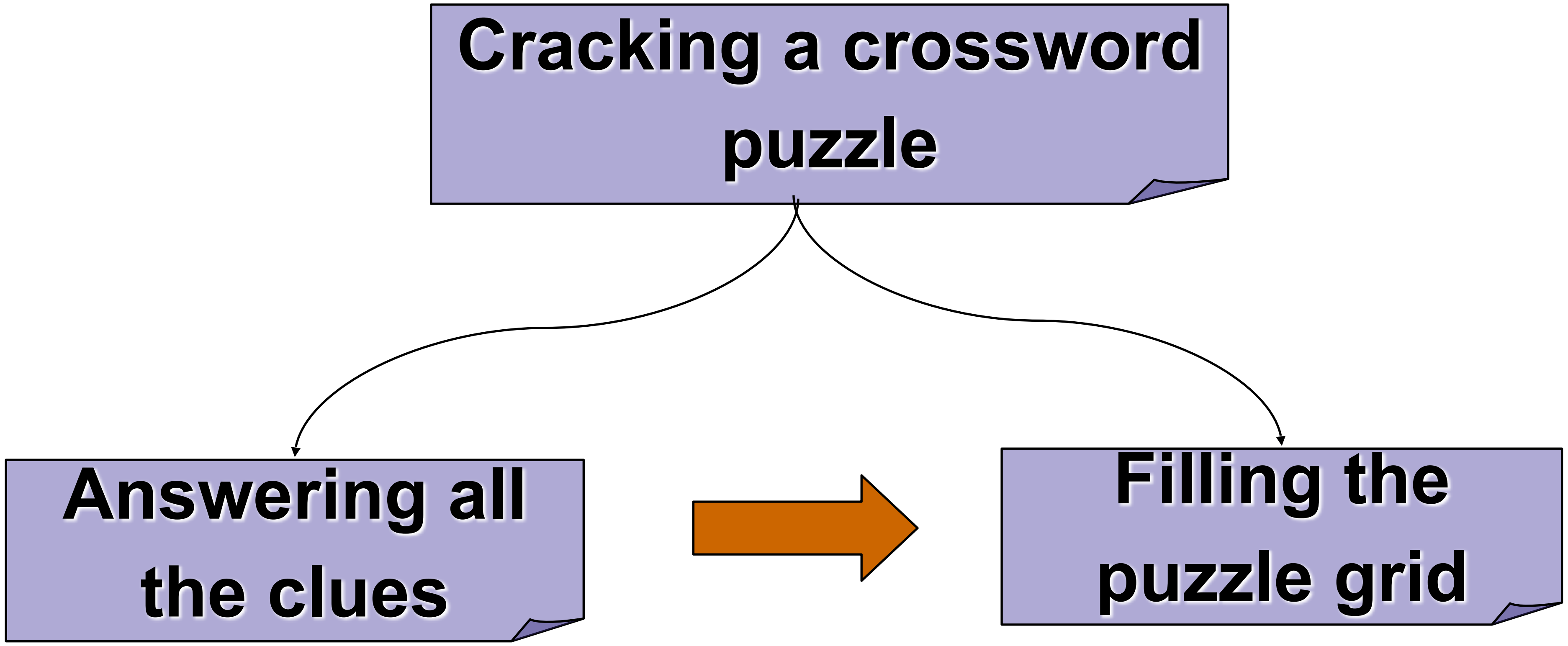
distance: number of tiles which are not in their correct positions



the greedy policy of following the minimum distance … and beyond!

2007
RAI 3 report on SAILAB challenge on Rubik's cube

CRACKING CROSSWORDS



Cracking a crossword puzzle

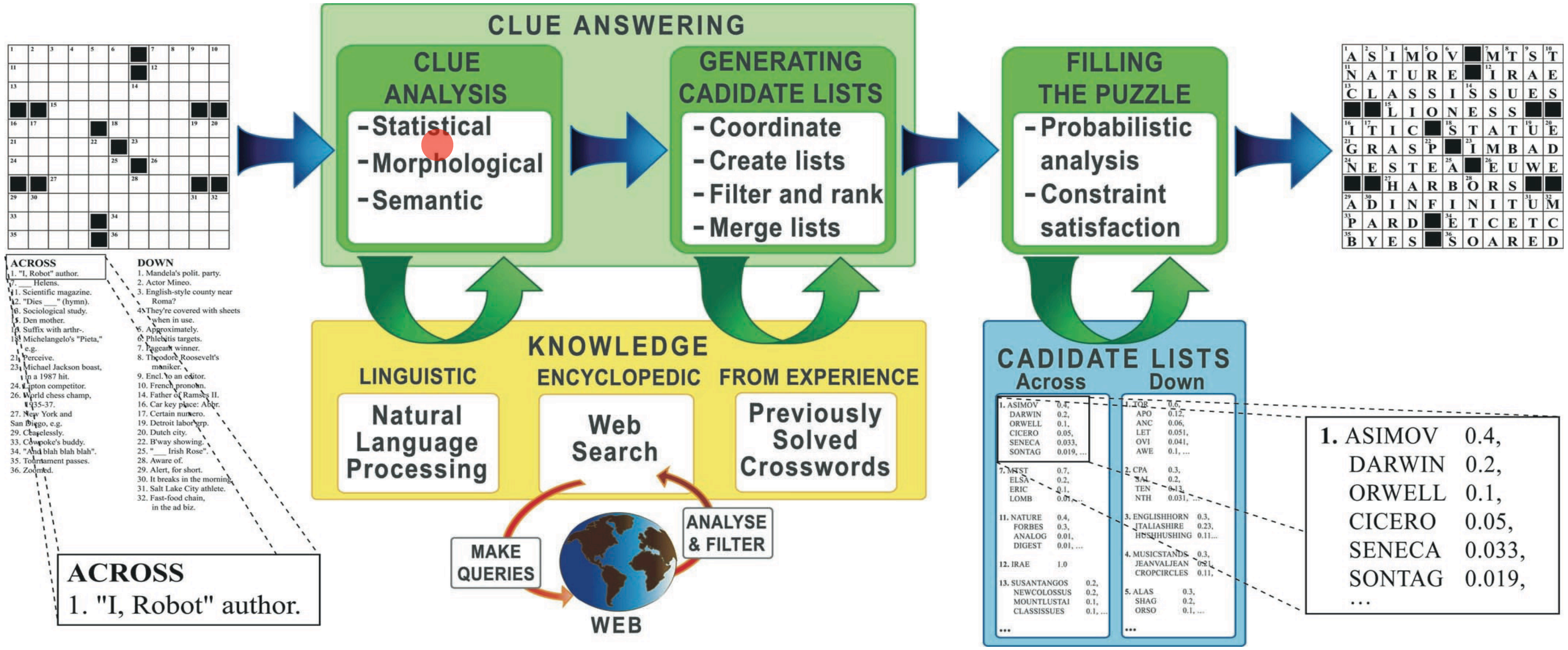
Answering all the clues

Filling the puzzle grid

Generating for each clue a long list of candidate answers

Putting the best combination of answers into the grid

WEBCROW @ SAILAB



ACROSS
1. "I, Robot" author.
7. ___ Helens.
11. Scientific magazine.
12. "Dies ___" (hymn).
13. Sociological study.
15. Den mother.
16. Suffix with arthr-.
18. Michelangelo's "Pieta," e.g.
21. Perceive.
23. Michael Jackson boast, in a 1987 hit.
24. Lipton competitor.
26. World chess champ, 1935-37.
27. New York and San Diego, e.g.
29. Ceaselessly.
33. Cowpoke's buddy.
34. "And blah blah blah".
35. Tournament passes.
36. Zoomed.

DOWN
1. Mandela's polit. party.
2. Actor Mineo.
3. English-style county near Roma?
4. They're covered with sheets when in use.
5. Approximately.
6. Phlebitis targets.
7. Pageant winner.
8. Theodore Roosevelt's maniker.
9. Encl. to an editor.
10. French pronoun.
14. Father of Ramses II.
16. Car key place: Abbr.
17. Certain numero.
19. Detroit labor grp.
20. Dutch city.
22. B'way showing.
25. "___ Irish Rose".
28. Aware of.
29. Alert, for short.
30. It breaks in the morning.
31. Salt Lake City athlete.
32. Fast-food chain, in the ad biz.

ACROSS
1. "I, Robot" author.

Across

1.	ASIMOV	0.4,
	DARWIN	0.2,
	ORWELL	0.1,
	CICERO	0.05,
	SENECA	0.033,
	SONTAG	0.019, ...
7.	MTST	0.7,
	ELSA	0.2,
	ERIC	0.1,
	LOMB	0.01, ...
11.	NATURE	0.4,
	FORBES	0.3,
	ANALOG	0.01,
	DIGEST	0.01, ...
12.	IRAE	1.0
13.	SUSANTANGOS	0.2,
	NEWCOLOSSUS	0.2,
	MOUNTLUSTAI	0.1,
	CLASSISSUES	0.1, ...
...		

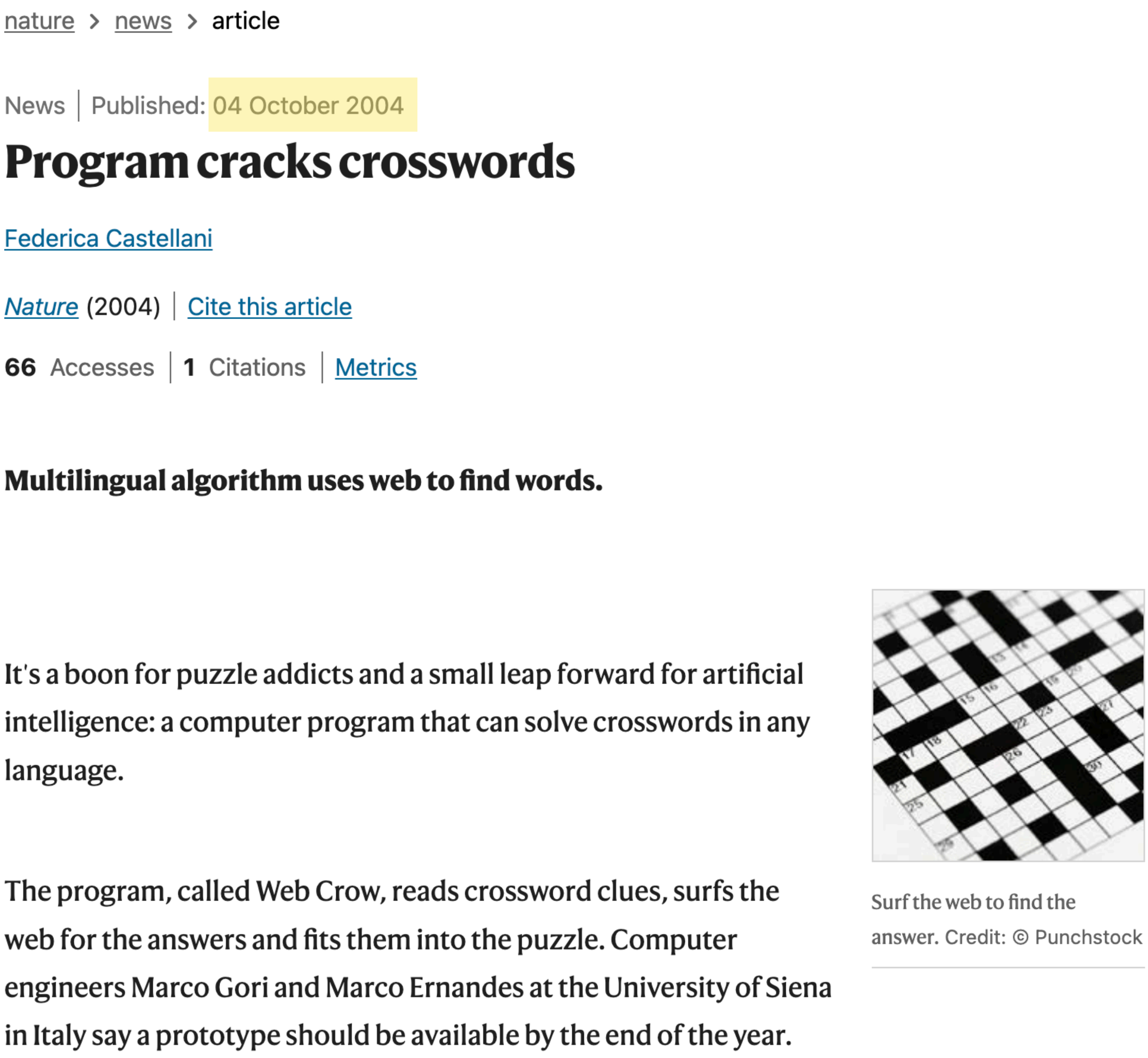
Down

1.	TOR	0.6,
	APO	0.12,
	ANC	0.06,
	LET	0.051,
	OVI	0.041,
	AWE	0.1, ...
2.	CPA	0.3,
	SAL	0.2,
	TEN	0.13,
	NTH	0.031, ...
3.	ENGLISHHORN	0.3,
	ITALIASHIRE	0.23,
	HUSHHUSHING	0.11...
4.	MUSICSTANDS	0.3,
	JEANVALJEAN	0.21,
	CROPCIRCLES	0.11,
5.	ALAS	0.3,
	SHAG	0.2,
	ORSO	0.1, ...
...		

1.	ASIMOV	0.4,
	DARWIN	0.2,
	ORWELL	0.1,
	CICERO	0.05,
	SENECA	0.033,
	SONTAG	0.019,
	...	

A	S	I	M	O	V	■	M	T	S	T
N	A	T	U	R	E	■	I	R	A	E
C	L	A	S	S	I	S	S	U	E	S
■	■	L	I	O	N	E	S	S	■	■
I	T	I	C	■	S	T	A	T	U	E
G	R	A	S	P	■	I	M	B	A	D
N	E	S	T	E	A	■	E	U	W	E
■	■	H	A	R	B	O	R	S	■	■
A	D	I	N	F	I	N	I	T	U	M
P	A	R	D	■	E	T	C	E	T	C
B	Y	E	S	■	S	O	A	R	E	D

S3P-2024 - The path of AI



nature > news > article

News | Published: 04 October 2004

Program cracks crosswords

Federica Castellani

Nature (2004) | Cite this article

66 Accesses | **1** Citations | Metrics

Multilingual algorithm uses web to find words.

It's a boon for puzzle addicts and a small leap forward for artificial intelligence: a computer program that can solve crosswords in any language.

The program, called Web Crow, reads crossword clues, surfs the web for the answers and fits them into the puzzle. Computer engineers Marco Gori and Marco Ernandes at the University of Siena in Italy say a prototype should be available by the end of the year.

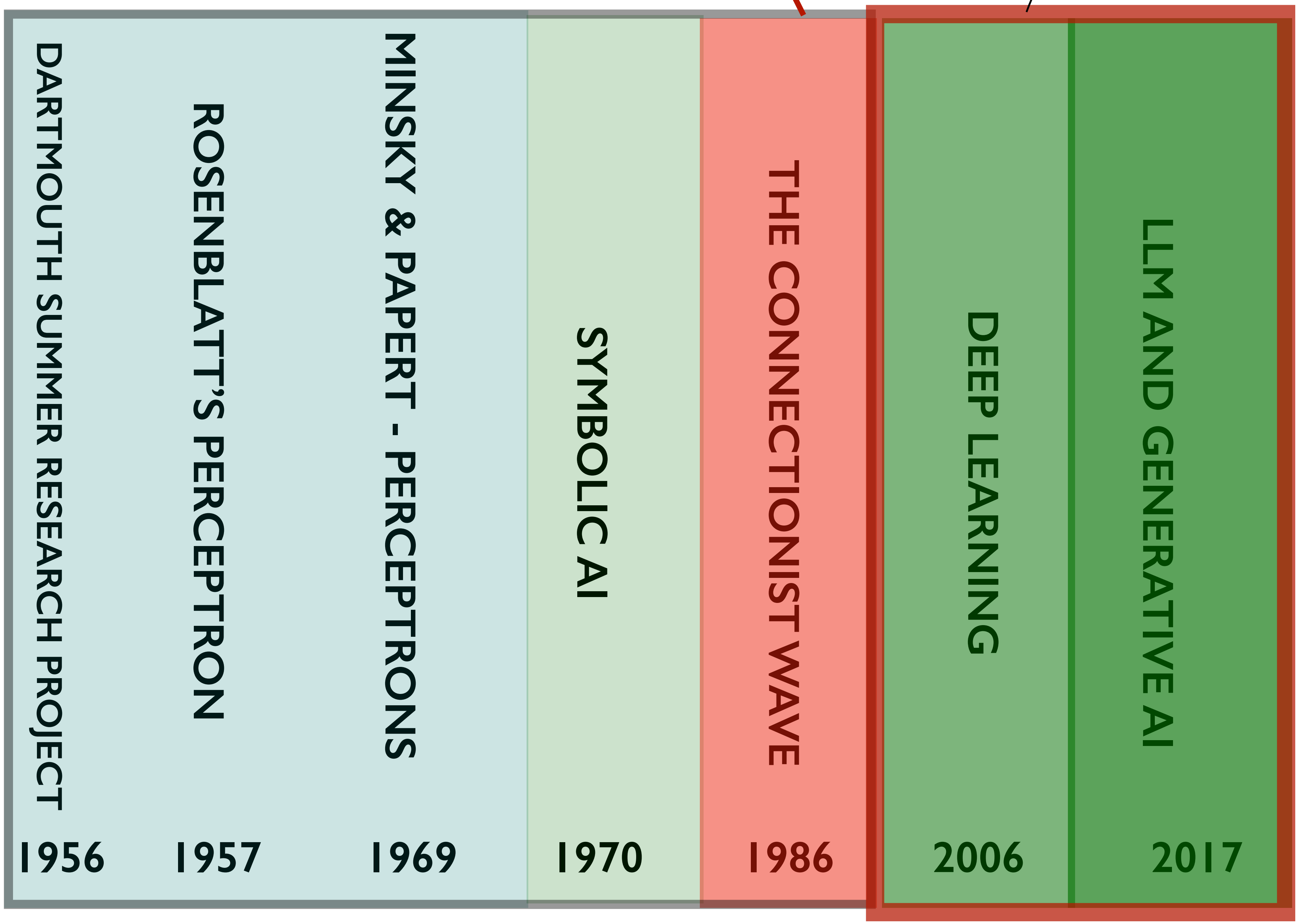
Surf the web to find the answer. Credit: © Punchstock

AI

THE BREAKTHROUGH OF CONNECTIONISM - PDP research group

the wave which supports AI

big collections and computational resources



DARTMOUTH SUMMER RESEARCH PROJECT 1956

ROSENBLATT'S PERCEPTRON 1957

MINSKY & PAPERT - PERCEPTRONS 1969

SYMBOLIC AI 1970

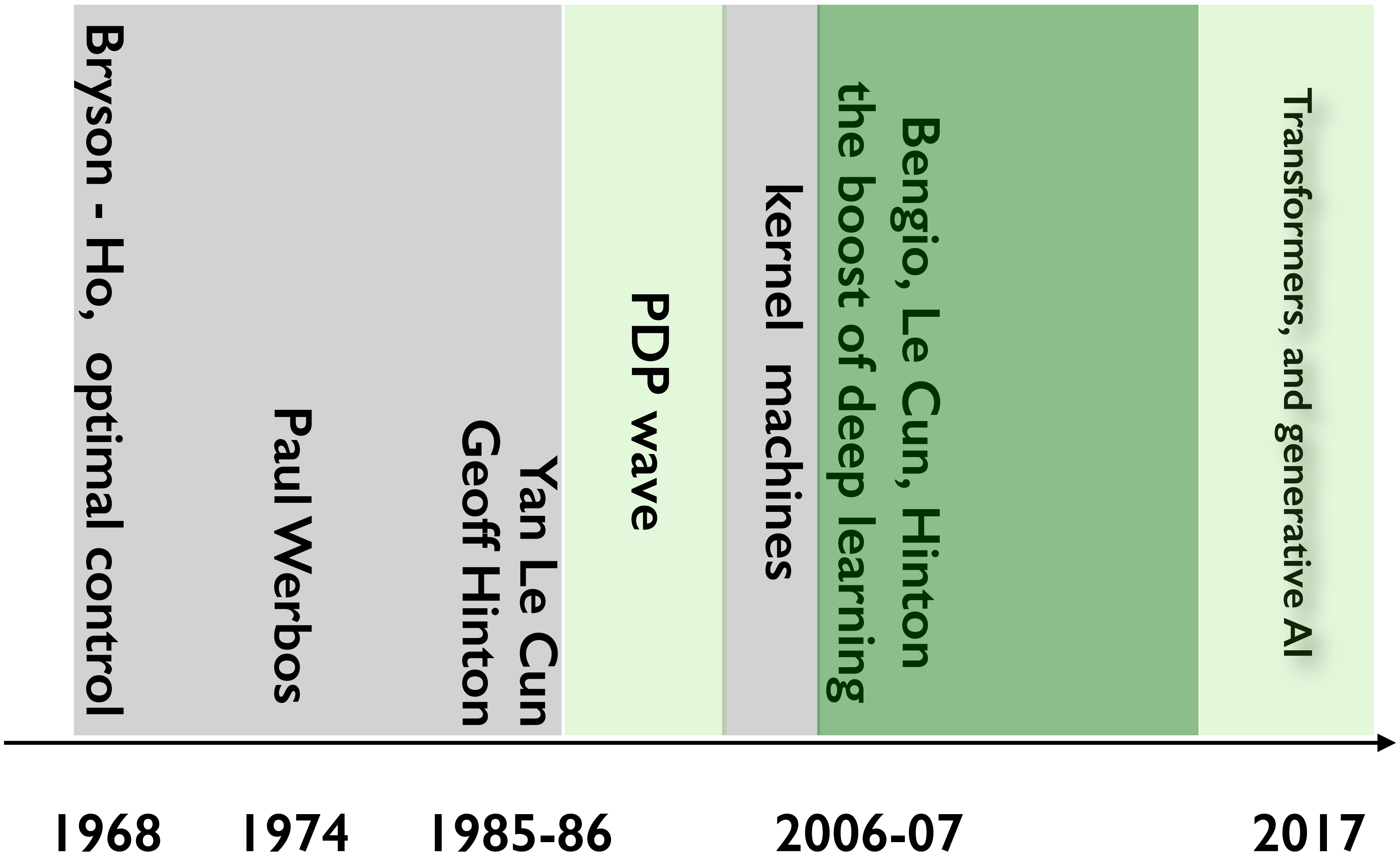
THE CONNECTIONIST WAVE 1986

DEEP LEARNING 2006

LLM AND GENERATIVE AI 2017

The Connectionist Wave

THE CHECKERED STORY OF BACKPROPAGATION AND BEYOND



Bryson - Ho, optimal control

Paul Werbos

Geoff Hinton
Yan Le Cun

PDP wave

kernel machines

the boost of deep learning
Bengio, Le Cun, Hinton

Transformers, and generative AI

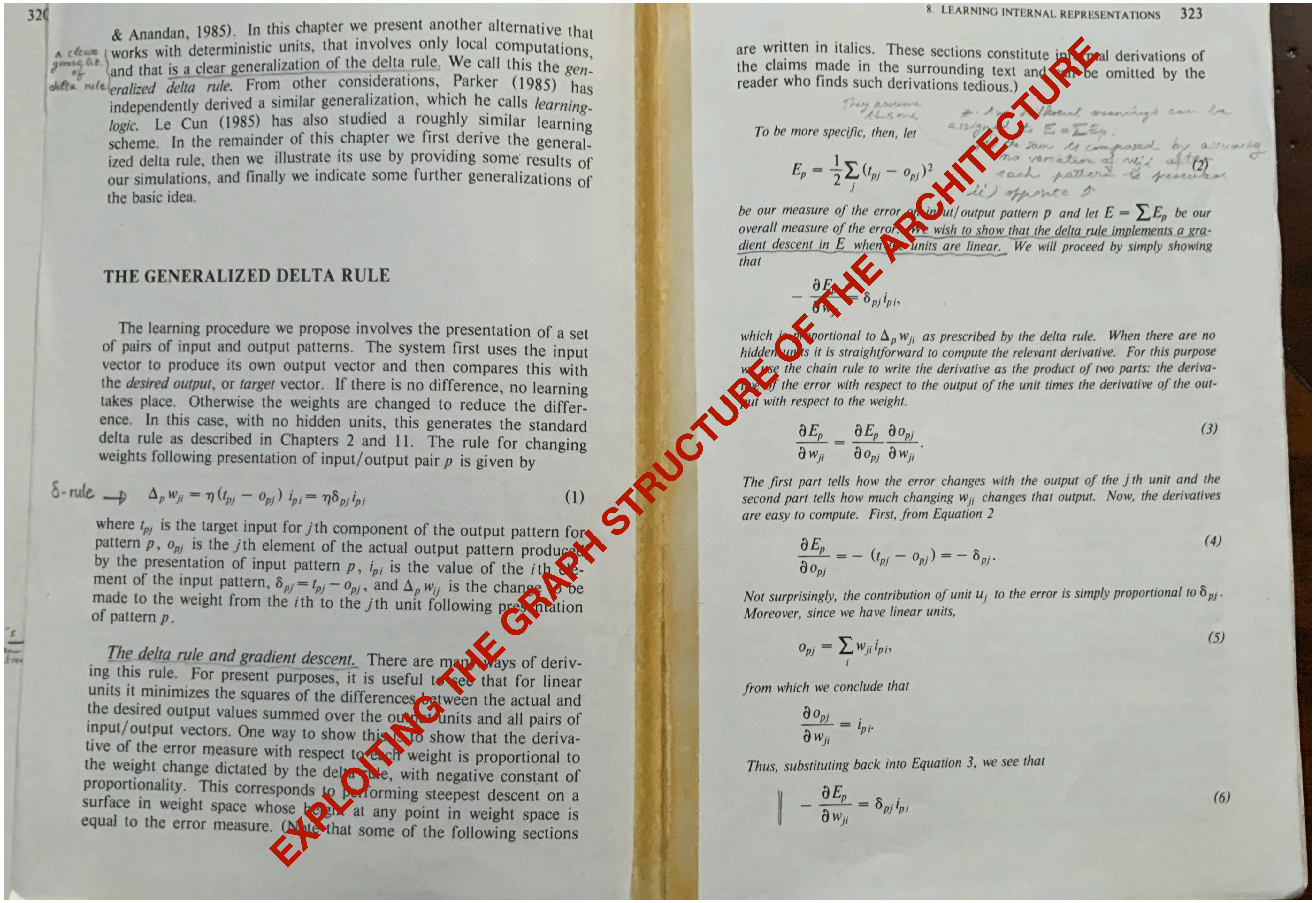
1968

1974

1985-86

2006-07

2017



320

& Anandan, 1985). In this chapter we present another alternative that works with deterministic units, that involves only local computations, and that is a clear generalization of the delta rule. We call this the *generalized delta rule.* From other considerations, Parker (1985) has independently derived a similar generalization, which he calls *learning-logic.* Le Cun (1985) has also studied a roughly similar learning scheme. In the remainder of this chapter we first derive the generalized delta rule, then we illustrate its use by providing some results of our simulations, and finally we indicate some further generalizations of the basic idea.

a clear generaliz. of delta rule

THE GENERALIZED DELTA RULE

The learning procedure we propose involves the presentation of a set of pairs of input and output patterns. The system first uses the input vector to produce its own output vector and then compares this with the *desired output*, or *target* vector. If there is no difference, no learning takes place. Otherwise the weights are changed to reduce the difference. In this case, with no hidden units, this generates the standard delta rule as described in Chapters 2 and 11. The rule for changing weights following presentation of input/output pair p is given by

δ-rule →
$$\Delta_p w_{ji} = \eta (t_{pj} - o_{pj})\, i_{pi} = \eta \delta_{pj} i_{pi} \qquad (1)$$

where t_{pj} is the target input for jth component of the output pattern for pattern p, o_{pj} is the jth element of the actual output pattern produced by the presentation of input pattern p, i_{pi} is the value of the ith element of the input pattern, $\delta_{pj} = t_{pj} - o_{pj}$, and $\Delta_p w_{ij}$ is the change to be made to the weight from the ith to the jth unit following presentation of pattern p.

The delta rule and gradient descent. There are many ways of deriving this rule. For present purposes, it is useful to see that for linear units it minimizes the squares of the differences between the actual and the desired output values summed over the output units and all pairs of input/output vectors. One way to show this is to show that the derivative of the error measure with respect to each weight is proportional to the weight change dictated by the delta rule, with negative constant of proportionality. This corresponds to performing steepest descent on a surface in weight space whose height at any point in weight space is equal to the error measure. (Note that some of the following sections

are written in italics. These sections constitute informal derivations of the claims made in the surrounding text and can be omitted by the reader who finds such derivations tedious.)

They assume that one

*: two different meanings can be assigned to E = ΣEp.
i) the sum is composed by assuming no variation of wji after each pattern is presented
ii) opposite

To be more specific, then, let

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \qquad (2)$$

be our measure of the error on input/output pattern p and let $E = \sum E_p$ be our overall measure of the error. We wish to show that the delta rule implements a gradient descent in E when the units are linear. We will proceed by simply showing that

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} i_{pi},$$

which is proportional to $\Delta_p w_{ji}$ as prescribed by the delta rule. When there are no hidden units it is straightforward to compute the relevant derivative. For this purpose we use the chain rule to write the derivative as the product of two parts: the derivative of the error with respect to the output of the unit times the derivative of the output with respect to the weight.

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ji}}. \qquad (3)$$

The first part tells how the error changes with the output of the jth unit and the second part tells how much changing w_{ji} changes that output. Now, the derivatives are easy to compute. First, from Equation 2

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}) = -\delta_{pj}. \qquad (4)$$

Not surprisingly, the contribution of unit u_j to the error is simply proportional to δ_{pj}. Moreover, since we have linear units,

$$o_{pj} = \sum_i w_{ji} i_{pi}, \qquad (5)$$

from which we conclude that

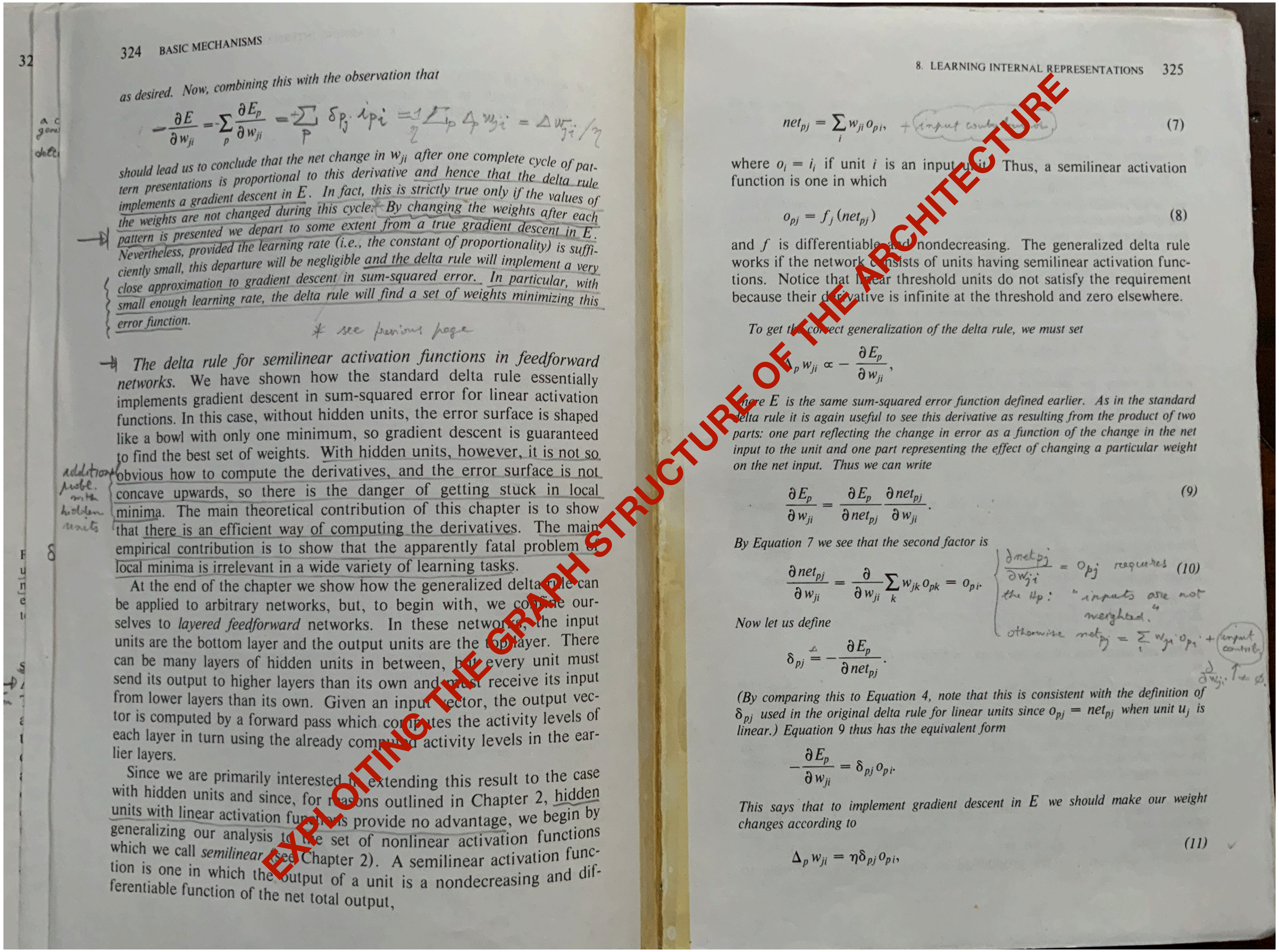
$$\frac{\partial o_{pj}}{\partial w_{ji}} = i_{pi}.$$

Thus, substituting back into Equation 3, we see that

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} i_{pi} \qquad (6)$$

EXPLOITING THE GRAPH STRUCTURE OF THE ARCHITECTURE

S3P-2024 - The connectionist wave



as desired. Now, combining this with the observation that

$$-\frac{\partial E}{\partial w_{ji}} = -\sum_p \frac{\partial E_p}{\partial w_{ji}} = \sum_p \delta_{pj} \cdot i_{pi} = \frac{1}{\eta} \sum_p \Delta_p w_{ji} = \Delta w_{ji}/\eta$$

should lead us to conclude that the net change in w_{ji} after one complete cycle of pattern presentations is proportional to this derivative and hence that the delta rule implements a gradient descent in E. In fact, this is strictly true only if the values of the weights are not changed during this cycle. By changing the weights after each pattern is presented we depart to some extent from a true gradient descent in E. Nevertheless, provided the learning rate (i.e., the constant of proportionality) is sufficiently small, this departure will be negligible and the delta rule will implement a very close approximation to gradient descent in sum-squared error. In particular, with small enough learning rate, the delta rule will find a set of weights minimizing this error function.*

* see previous page

The delta rule for semilinear activation functions in feedforward networks. We have shown how the standard delta rule essentially implements gradient descent in sum-squared error for linear activation functions. In this case, without hidden units, the error surface is shaped like a bowl with only one minimum, so gradient descent is guaranteed to find the best set of weights. With hidden units, however, it is not so obvious how to compute the derivatives, and the error surface is not concave upwards, so there is the danger of getting stuck in local minima. The main theoretical contribution of this chapter is to show that there is an efficient way of computing the derivatives. The main empirical contribution is to show that the apparently fatal problem of local minima is irrelevant in a wide variety of learning tasks.

additional prob. with hidden units

At the end of the chapter we show how the generalized delta rule can be applied to arbitrary networks, but, to begin with, we confine ourselves to *layered feedforward* networks. In these networks, the input units are the bottom layer and the output units are the top layer. There can be many layers of hidden units in between, but every unit must send its output to higher layers than its own and must receive its input from lower layers than its own. Given an input vector, the output vector is computed by a forward pass which computes the activity levels of each layer in turn using the already computed activity levels in the earlier layers.

Since we are primarily interested in extending this result to the case with hidden units and since, for reasons outlined in Chapter 2, hidden units with linear activation functions provide no advantage, we begin by generalizing our analysis to the set of nonlinear activation functions which we call *semilinear* (see Chapter 2). A semilinear activation function is one in which the output of a unit is a nondecreasing and differentiable function of the net total output,

$$net_{pj} = \sum_i w_{ji} o_{pi}, \quad + \text{input contribution} \tag{7}$$

where $o_i = i_i$ if unit i is an input unit. Thus, a semilinear activation function is one in which

$$o_{pj} = f_j(net_{pj}) \tag{8}$$

and f is differentiable and nondecreasing. The generalized delta rule works if the network consists of units having semilinear activation functions. Notice that linear threshold units do not satisfy the requirement because their derivative is infinite at the threshold and zero elsewhere.

To get the correct generalization of the delta rule, we must set

$$\Delta_p w_{ji} \propto -\frac{\partial E_p}{\partial w_{ji}},$$

where E is the same sum-squared error function defined earlier. As in the standard delta rule it is again useful to see this derivative as resulting from the product of two parts: one part reflecting the change in error as a function of the change in the net input to the unit and one part representing the effect of changing a particular weight on the net input. Thus we can write

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ji}}. \tag{9}$$

By Equation 7 we see that the second factor is

$$\frac{\partial net_{pj}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_k w_{jk} o_{pk} = o_{pi}. \tag{10}$$

$\frac{\partial net_{pj}}{\partial w_{ji}} = o_{pj}$ requires the Hp: "inputs are not weighted."
otherwise $net_{pj} = \sum_i w_{ji} o_{pi} + \text{input contrib.}$, $\frac{\partial}{\partial w_{ji}} \uparrow \neq \emptyset$.

Now let us define

$$\delta_{pj} \overset{\Delta}{=} -\frac{\partial E_p}{\partial net_{pj}}.$$

(By comparing this to Equation 4, note that this is consistent with the definition of δ_{pj} used in the original delta rule for linear units since $o_{pj} = net_{pj}$ when unit u_j is linear.) Equation 9 thus has the equivalent form

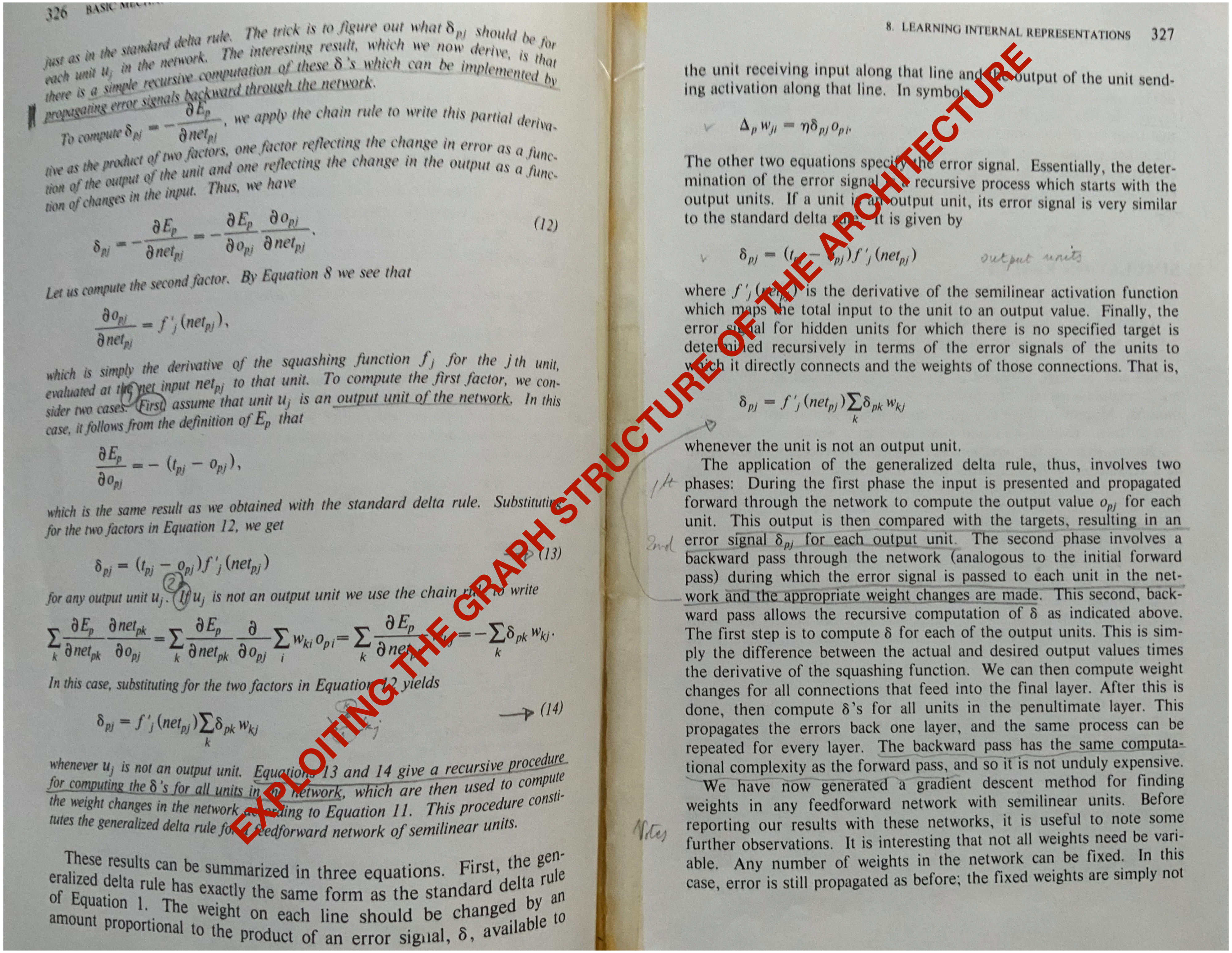
$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} o_{pi}.$$

This says that to implement gradient descent in E we should make our weight changes according to

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi}, \tag{11}$$

EXPLOITING THE GRAPH STRUCTURE OF THE ARCHITECTURE

S3P-2024 - The connectionist wave



EXPLOITING THE GRAPH STRUCTURE OF THE ARCHITECTURE

326 BASIC MEC

just as in the standard delta rule. The trick is to figure out what δ_{pj} should be for each unit u_j in the network. The interesting result, which we now derive, is that there is a simple recursive computation of these δ's which can be implemented by propagating error signals backward through the network.

To compute $\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}}$, we apply the chain rule to write this partial derivative as the product of two factors, one factor reflecting the change in error as a function of the output of the unit and one reflecting the change in the output as a function of changes in the input. Thus, we have

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}}. \tag{12}$$

Let us compute the second factor. By Equation 8 we see that

$$\frac{\partial o_{pj}}{\partial net_{pj}} = f'_j(net_{pj}),$$

which is simply the derivative of the squashing function f_j for the jth unit, evaluated at the net input net_{pj} to that unit. To compute the first factor, we consider two cases. 1 First, assume that unit u_j is an output unit of the network. In this case, it follows from the definition of E_p that

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}),$$

which is the same result as we obtained with the standard delta rule. Substituting for the two factors in Equation 12, we get

$$\delta_{pj} = (t_{pj} - o_{pj}) f'_j(net_{pj}) \tag{13}$$

for any output unit u_j. 2 If u_j is not an output unit we use the chain rule to write

$$\sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial o_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial}{\partial o_{pj}} \sum_i w_{ki} o_{pi} = \sum_k \frac{\partial E_p}{\partial net_{pk}} w_{kj} = -\sum_k \delta_{pk} w_{kj}.$$

In this case, substituting for the two factors in Equation 12 yields

$$\delta_{pj} = f'_j(net_{pj}) \sum_k \delta_{pk} w_{kj} \tag{14}$$

whenever u_j is not an output unit. Equations 13 and 14 give a recursive procedure for computing the δ's for all units in the network, which are then used to compute the weight changes in the network according to Equation 11. This procedure constitutes the generalized delta rule for a feedforward network of semilinear units.

These results can be summarized in three equations. First, the generalized delta rule has exactly the same form as the standard delta rule of Equation 1. The weight on each line should be changed by an amount proportional to the product of an error signal, δ, available to

8. LEARNING INTERNAL REPRESENTATIONS 327

the unit receiving input along that line and the output of the unit sending activation along that line. In symbols,

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi}.$$

The other two equations specify the error signal. Essentially, the determination of the error signal is a recursive process which starts with the output units. If a unit is an output unit, its error signal is very similar to the standard delta rule. It is given by

$$\delta_{pj} = (t_{pj} - o_{pj}) f'_j(net_{pj})$$

output units

where $f'_j(net_{pj})$ is the derivative of the semilinear activation function which maps the total input to the unit to an output value. Finally, the error signal for hidden units for which there is no specified target is determined recursively in terms of the error signals of the units to which it directly connects and the weights of those connections. That is,

$$\delta_{pj} = f'_j(net_{pj}) \sum_k \delta_{pk} w_{kj}$$

whenever the unit is not an output unit.

The application of the generalized delta rule, thus, involves two phases: During the first phase the input is presented and propagated forward through the network to compute the output value o_{pj} for each unit. This output is then compared with the targets, resulting in an error signal δ_{pj} for each output unit. The second phase involves a backward pass through the network (analogous to the initial forward pass) during which the error signal is passed to each unit in the network and the appropriate weight changes are made. This second, backward pass allows the recursive computation of δ as indicated above. The first step is to compute δ for each of the output units. This is simply the difference between the actual and desired output values times the derivative of the squashing function. We can then compute weight changes for all connections that feed into the final layer. After this is done, then compute δ's for all units in the penultimate layer. This propagates the errors back one layer, and the same process can be repeated for every layer. The backward pass has the same computational complexity as the forward pass, and so it is not unduly expensive.

1st

2nd

We have now generated a gradient descent method for finding weights in any feedforward network with semilinear units. Before reporting our results with these networks, it is useful to note some further observations. It is interesting that not all weights need be variable. Any number of weights in the network can be fixed. In this case, error is still propagated as before; the fixed weights are simply not

Notes

GRADIENT

Why not to use classic numerical schemes?

complexity

$O(M^2)$

number of weights

two-point based approximation

$$\frac{\partial e}{\partial w_{ij}} \leftarrow \frac{e(w_{ij}+h, v, y) - e(w_{ij}+h, v, y)}{2h}$$

four-point based approximation

$$\sigma^{(1)}(a) = \frac{-\sigma(a+2h) + 8\sigma(a+h) - 8\sigma(a-h) + \sigma(a-2h)}{12h} + \frac{h^4}{30}\sigma^{(5)}(\tilde{a})$$

THE “SECRET” OF BACKPROPAGATION

$$\frac{\partial e}{\partial w} = \frac{\partial V}{\partial f} \cdot \frac{\partial f}{\partial w} = \underbrace{\sum_{o \in \mathscr{O}} \frac{\partial V}{\partial f_o} \frac{\partial f_o}{\partial w}}_{\text{outputs}}.$$

THE CHAIN RULE

backward term

$$g_{ij}^{o} = \frac{\partial x_o}{\partial w_{ij}} = \frac{\partial x_o}{\partial a_i}\frac{\partial a_i}{\partial w_{ij}} = \frac{\partial x_o}{\partial a_i}\frac{\partial}{\partial w_{ij}}\sum_{h \in \mathrm{pa}(i)} w_{ih} x_h = \delta_i^o x_j$$

forward term

$$\delta_i^o := \partial x_o / \partial a_i$$

$$\delta_o^o = \sigma'(a_o)$$

Backward step

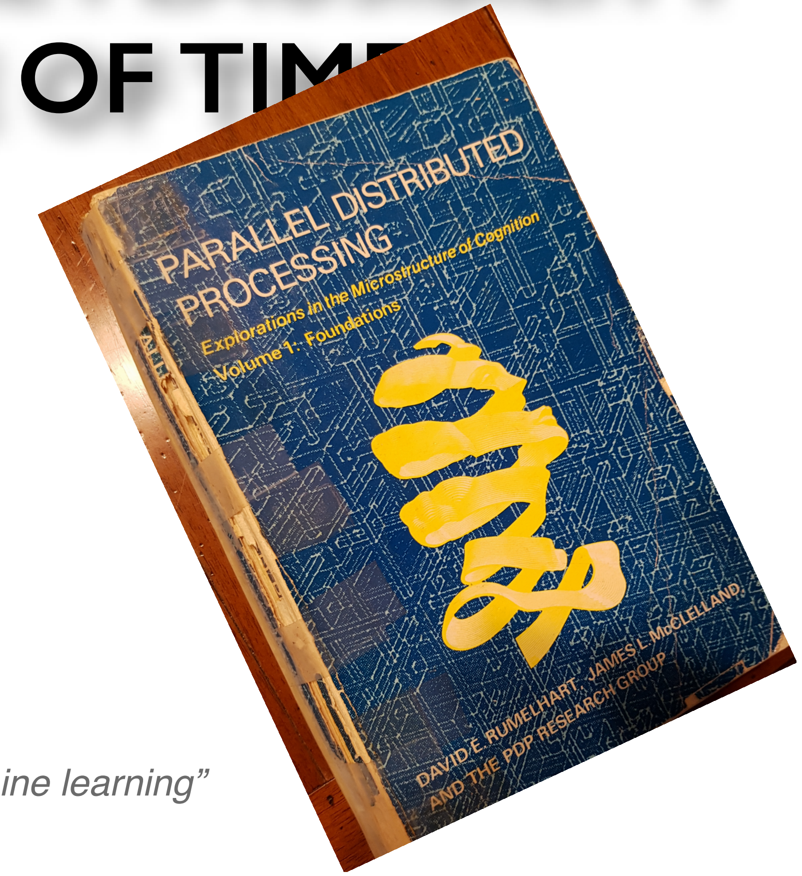
$$\delta_i^o = \frac{\partial x_o}{\partial a_i} = \sum_{h \in \mathrm{ch}(i)} \frac{\partial x_o}{\partial a_h} \frac{\partial a_h}{\partial x_i} \frac{\partial x_i}{\partial a_i} = \sigma'(a_i) \sum_{h \in \mathrm{ch}(i)} w_{hi} \delta_h^o$$

$$\frac{\partial V}{\partial w_{ij}} = \frac{\partial V}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} = \delta_i x_j$$

"magic factorization": this is why we have $O(M)$ floating-point operation!

What about computing the gradient with billion of weights?
$10^9 \quad vs \quad 10^{18}$

BP, BIOLOGICAL PLAUSIBILITY AND THE ROLE OF TIM


PARALLEL DISTRIBUTED PROCESSING
Explorations in the Microstructure of Cognition
Volume 1: Foundations
DAVID E. RUMELHART, JAMES L. McCLELLAND, AND THE PDP RESEARCH GROUP

NeurIPS 2020 Workshop on "Beyond Backpropagation"
LOD 2020 Workshop on Biologically Plausible Learning

14:55- 15:00 **Alessandro Sperduti**,"Introduction"

15:00 – 15:30 **Tomaso Poggio**, *"Towards new foundations for machine learning"*

15:30 – 16:00 **Yoshua Bengio**, *"Equilibrium Propagation"*

16:00 – 16:30 **Naftali Tishby**, *"Local Information Bottleneck optimization as a Biologically plausible feedforward learning mechanism"*

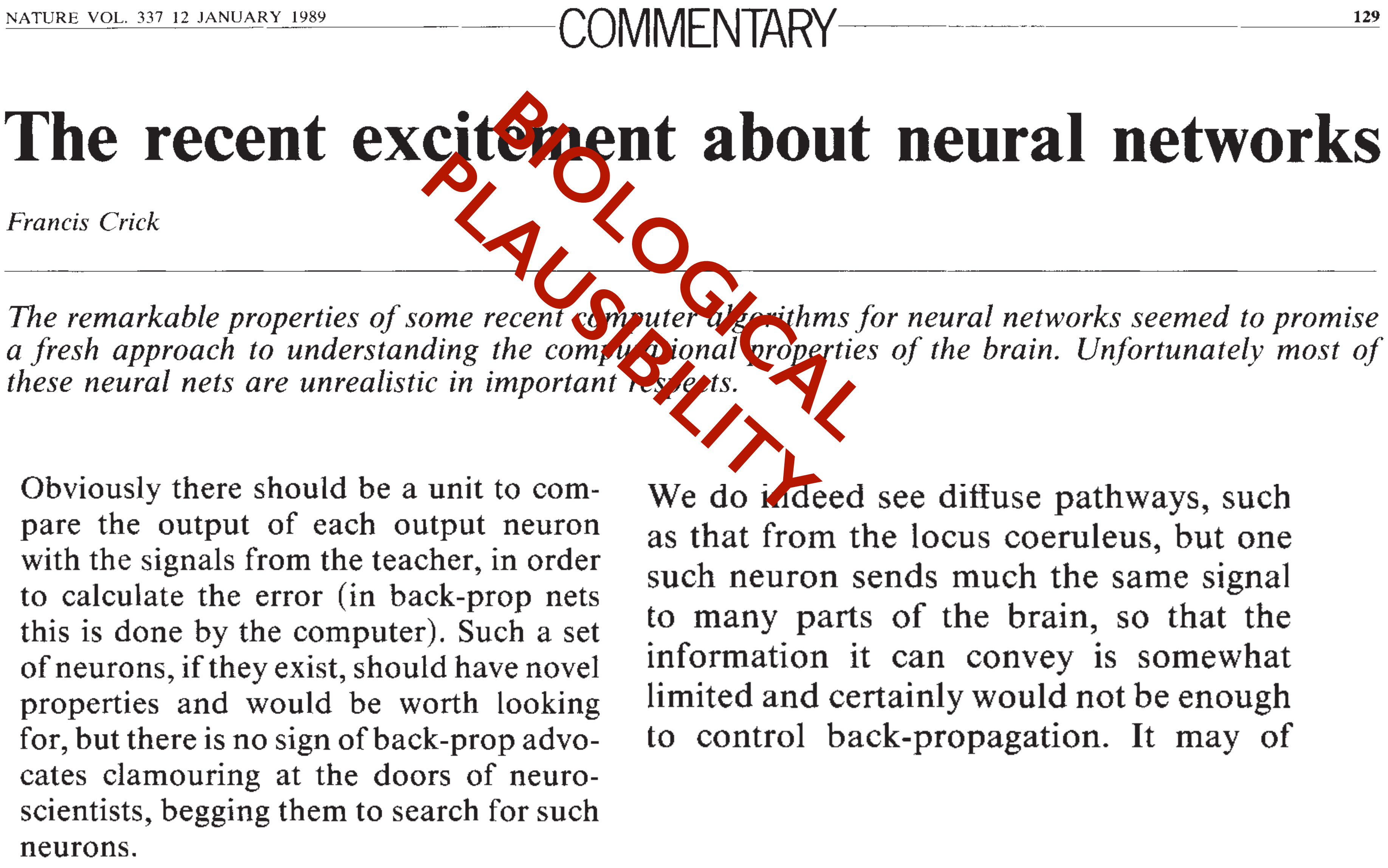
Coffee Break

16:45-17:15 **Pierre Baldi**, *"The Theory of Local Learning"*

17:15 – 17:45 **Marco Gori**, *"Backprop Diffusion is Locally Plausible"*

17:45 – 18:15 **Cristina Savin**, *"TBA"*

Francis Crick, 1989



NATURE VOL. 337 12 JANUARY 1989 — COMMENTARY — 129

The recent excitement about neural networks

Francis Crick

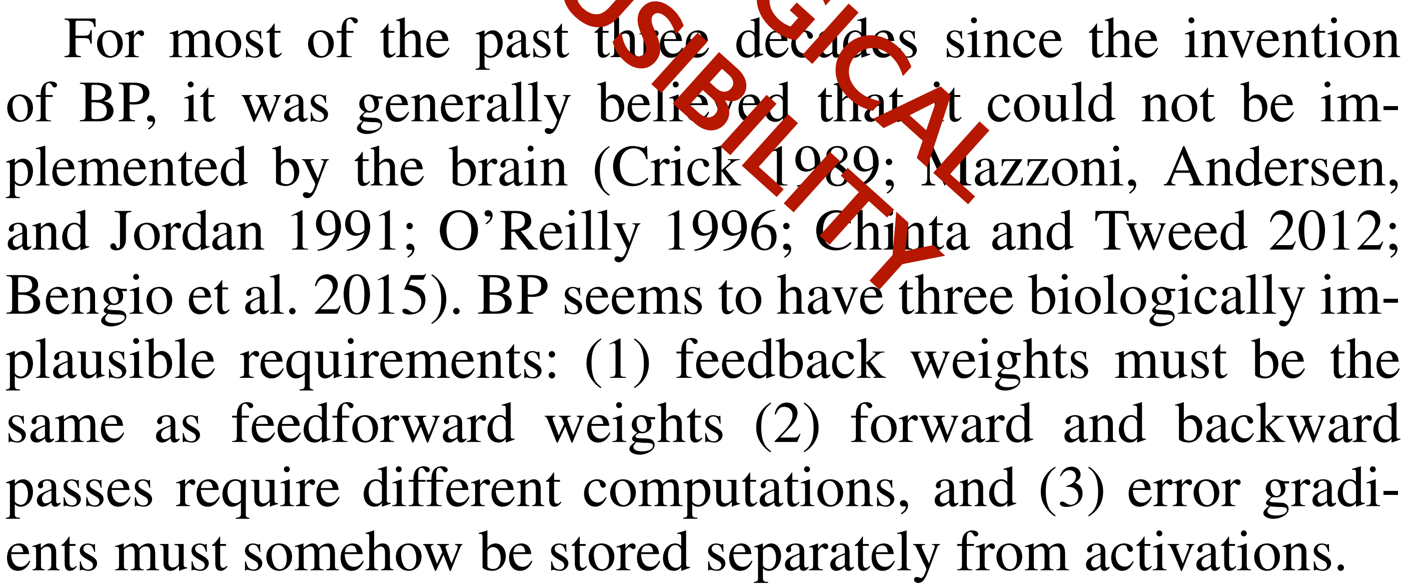
The remarkable properties of some recent computer algorithms for neural networks seemed to promise a fresh approach to understanding the computational properties of the brain. Unfortunately most of these neural nets are unrealistic in important respects.

Obviously there should be a unit to compare the output of each output neuron with the signals from the teacher, in order to calculate the error (in back-prop nets this is done by the computer). Such a set of neurons, if they exist, should have novel properties and would be worth looking for, but there is no sign of back-prop advocates clamouring at the doors of neuroscientists, begging them to search for such neurons.

We do indeed see diffuse pathways, such as that from the locus coeruleus, but one such neuron sends much the same signal to many parts of the brain, so that the information it can convey is somewhat limited and certainly would not be enough to control back-propagation. It may of

BIOLOGICAL PLAUSIBILITY

Tomaso Poggio et al, 2016



For most of the past three decades since the invention of BP, it was generally believed that it could not be implemented by the brain (Crick 1989; Mazzoni, Andersen, and Jordan 1991; O'Reilly 1996; Chinta and Tweed 2012; Bengio et al. 2015). BP seems to have three biologically implausible requirements: (1) feedback weights must be the same as feedforward weights (2) forward and backward passes require different computations, and (3) error gradients must somehow be stored separately from activations.

BIOLOGICAL PLAUSIBILITY

BIOLOGICAL PLAUSIBILITY OF BACKPROPAGATION

While argue about BP biological plausibility (the straw)
but we should first argue about static (the beam) neural models!

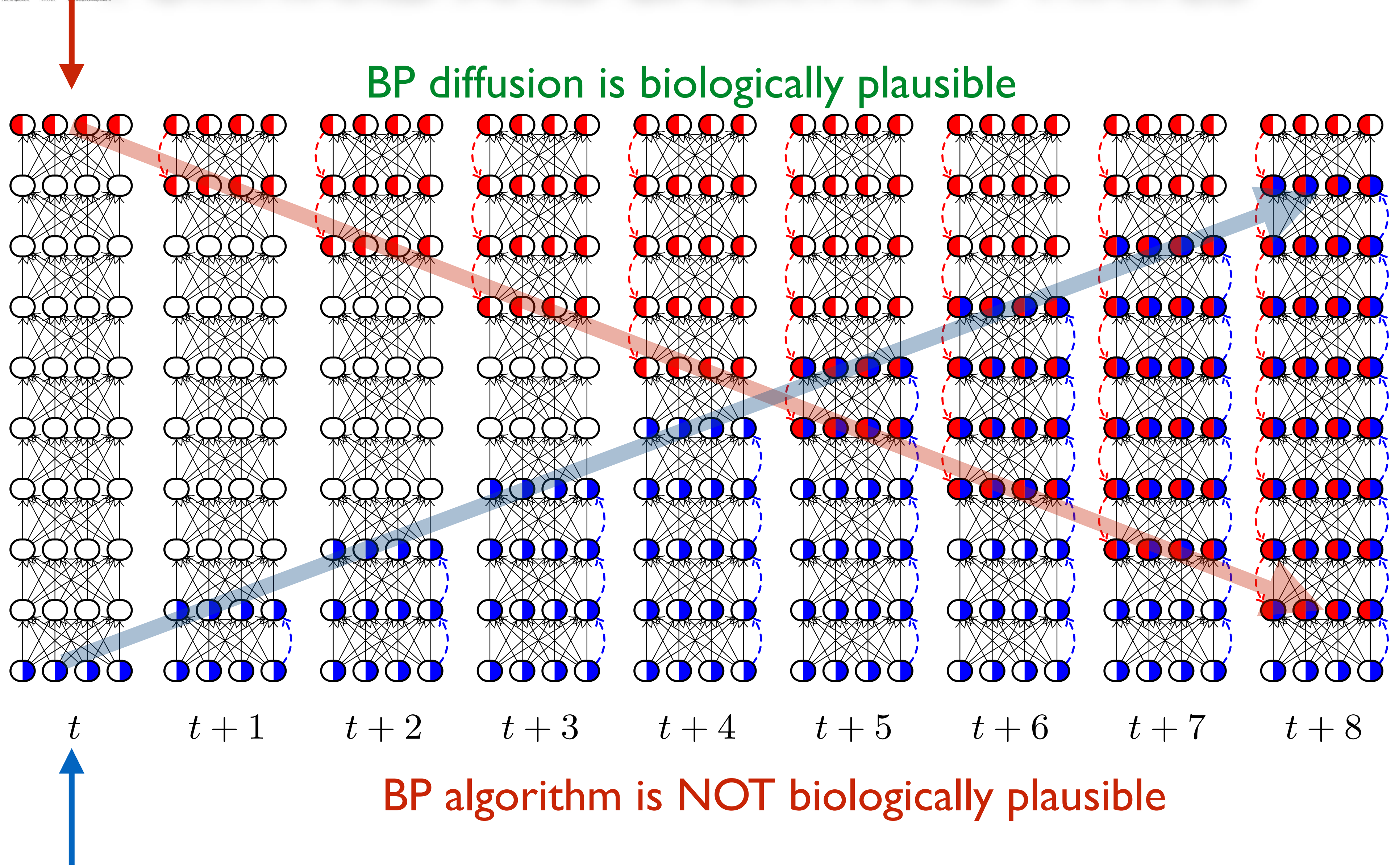
$$x = \sigma\Big(\sum_{\kappa} w_{\kappa} x_{\kappa}\Big)$$

Where is time? Causality?

We are mostly missing physical plausibility!

Reframing learning in time leads to conquer
higher degree of autonomy and consciousness

FORWARD AND BACKWARD WAVES



BP diffusion is biologically plausible

BP algorithm is NOT biologically plausible

COLLECTIONLESS AI

THE PROGRESS ON WEB SEARCH

Ian H. Witten • Marco Gori • Teresa Numerico
WEB
DRAGONS
Inside the Myths of Search Engine Technology
DISCARD

In oriental folklore dragons not only enjoy awesome grace and beauty, they are endowed with immense wisdom. But in the west they are often portrayed as evil—St. George vanquishes a fearsome dragon, as does Beowulf—or, sometimes, friendly—Puff.

ANY PROBLEM?

- Web dragons surveils the treasure
- The secret paradox
- Privacy issues

GENERATIVE AI: DRAGONS

The milk: it is not just
Web search or GMail …

COLLECTIONLESS CHALLENGES

Under control of public bodies

Design Machines which acquire intelligence
solely by environmental interactions without
recording the processed information

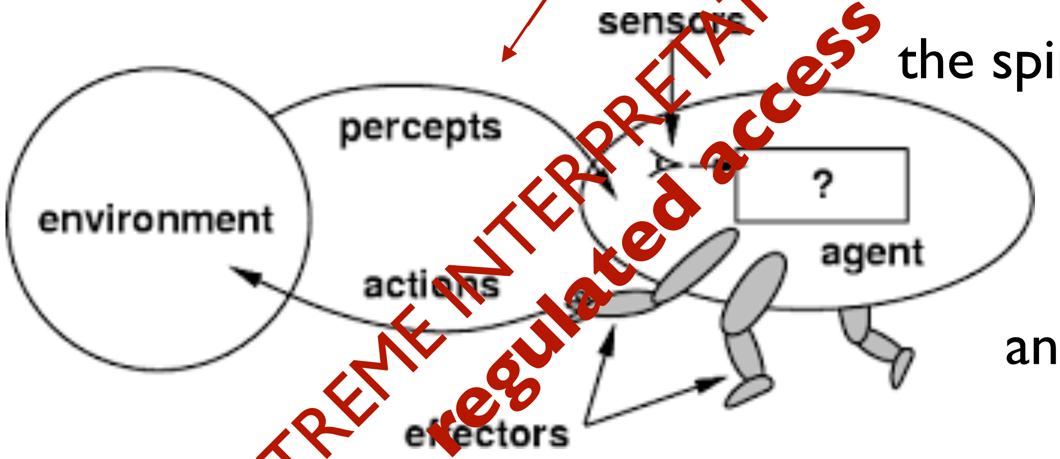
EXTREME INTERPRETATION OF LIFE LONG LEARNING

regulated access to public databases

Beyond
the spirit of reinforcement learning

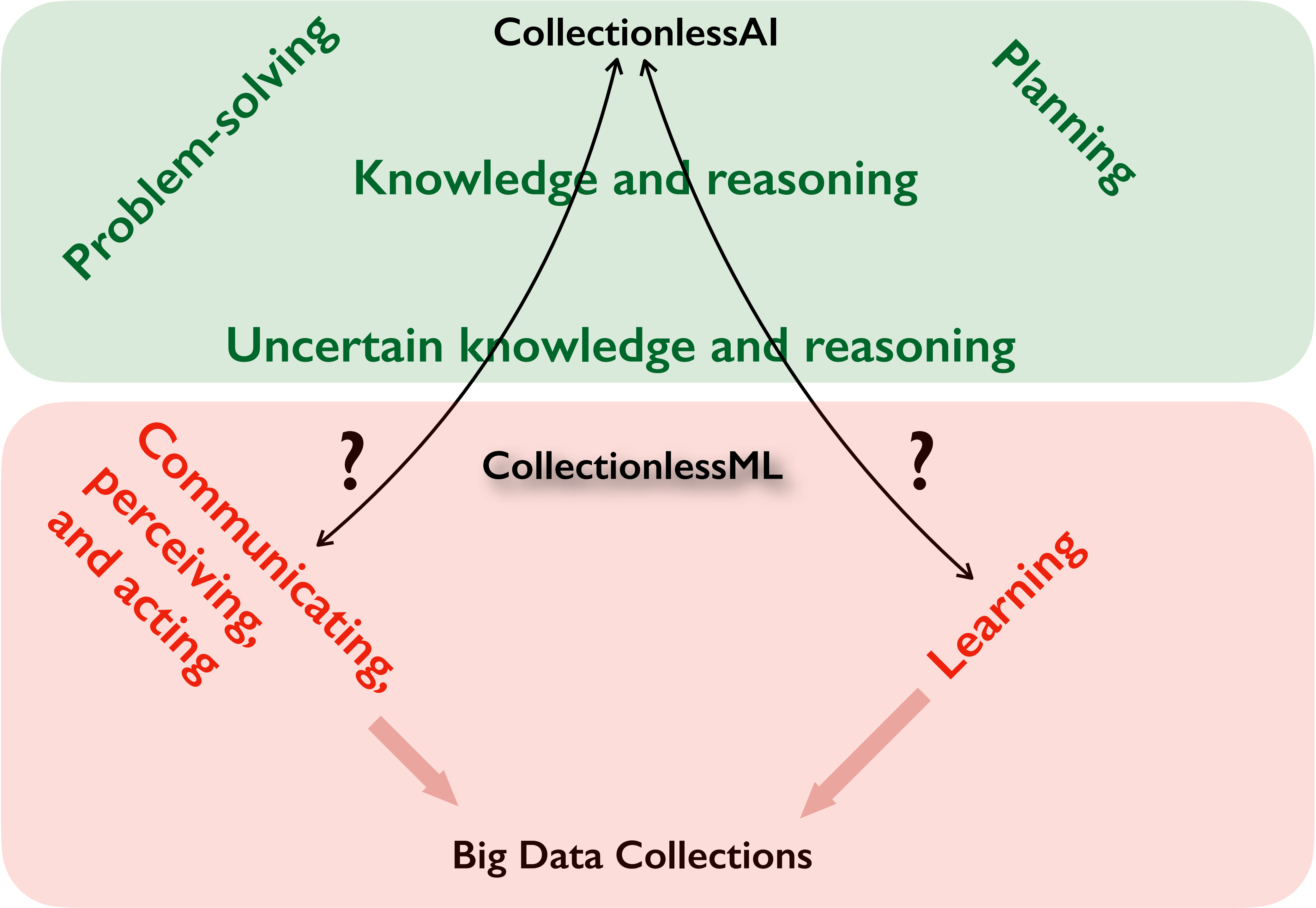
↓

environmental interactions
and the need for conscious actions



AI

According to Norvig-Russell's textbook



CollectionlessAI

Problem-solving

Planning

Knowledge and reasoning

Uncertain knowledge and reasoning

Communicating,
perceiving,
and acting

?

CollectionlessML

?

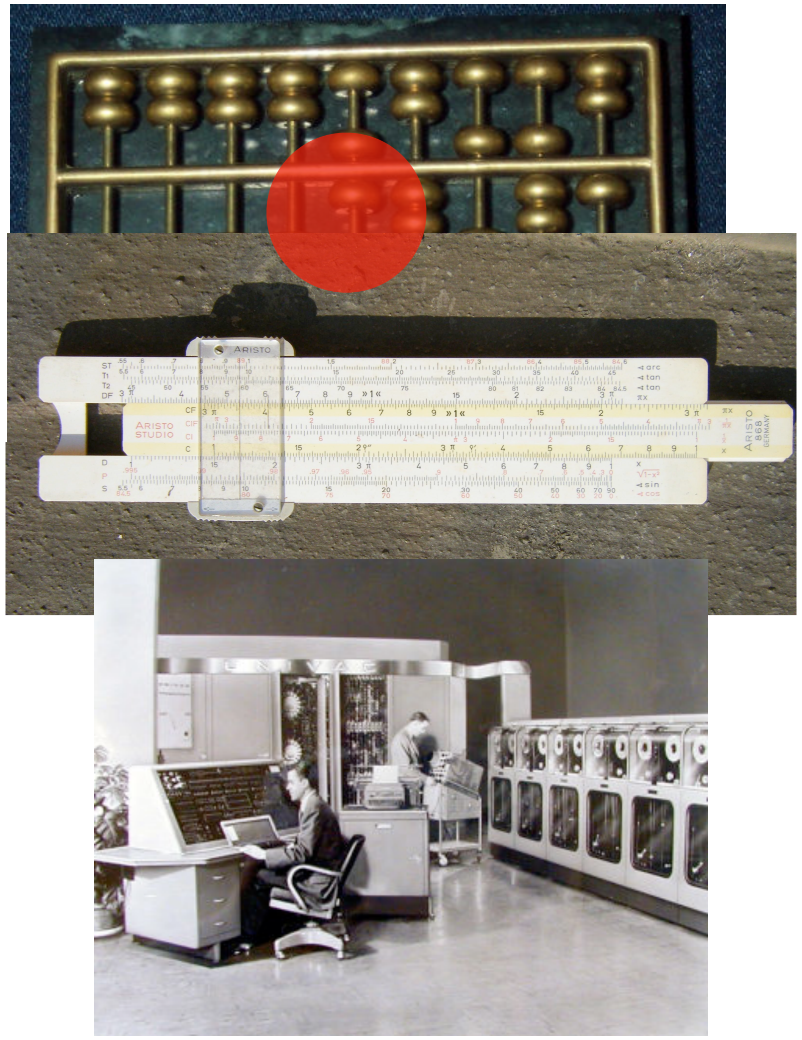
Learning

Big Data Collections

The Challenges Of The Transition From

Ai to aI

SMELLING THE HISTORY OF COMPUTATION
from slide rule to smartphone



S3P-2024 - Collectionless AI

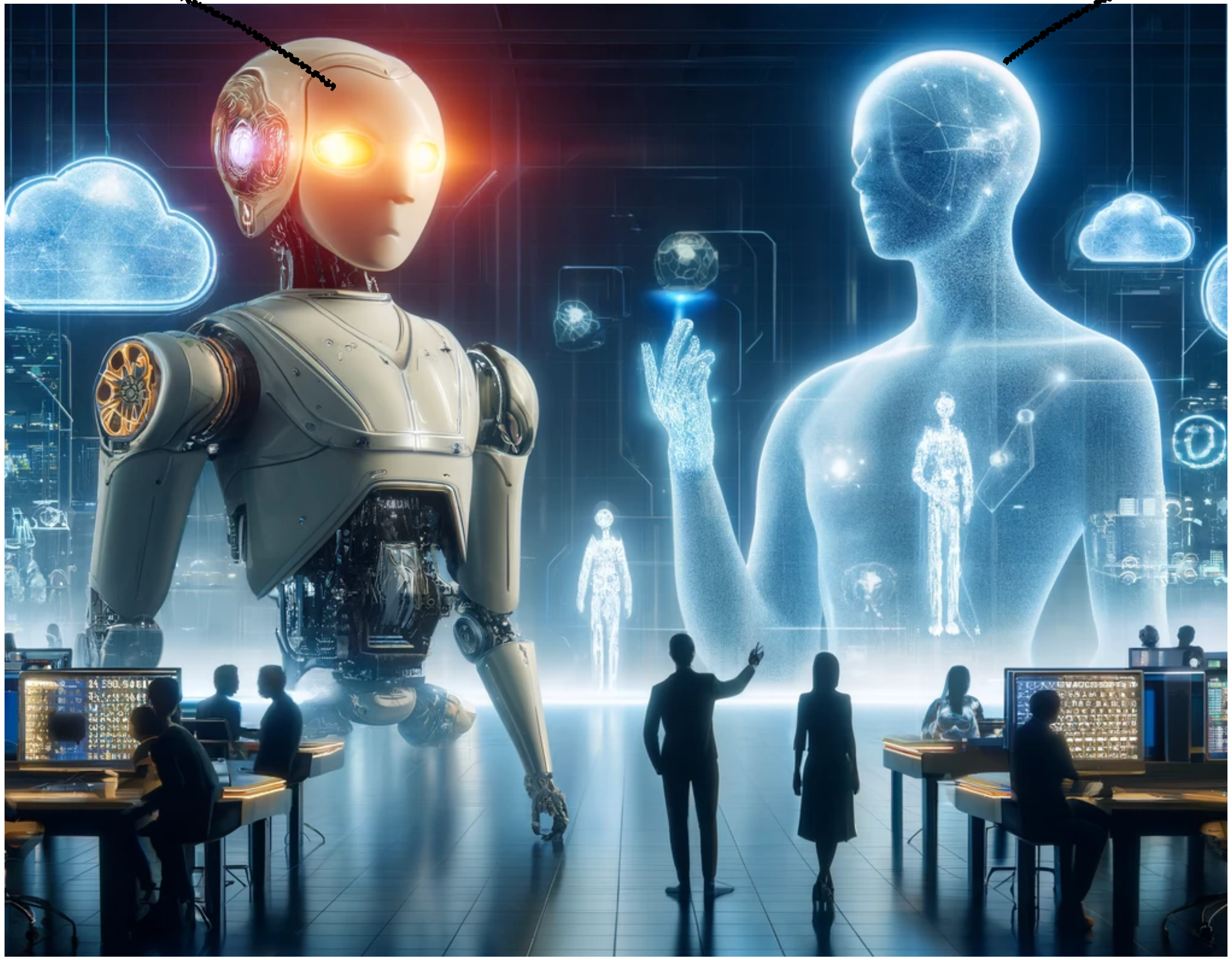
COLLECTIONLESS AI

CAI

On board intelligence

Collectionless-based Machine Intelligence
- birth
- life
- reproduction
- death

transfer learning …
evolutionary computation



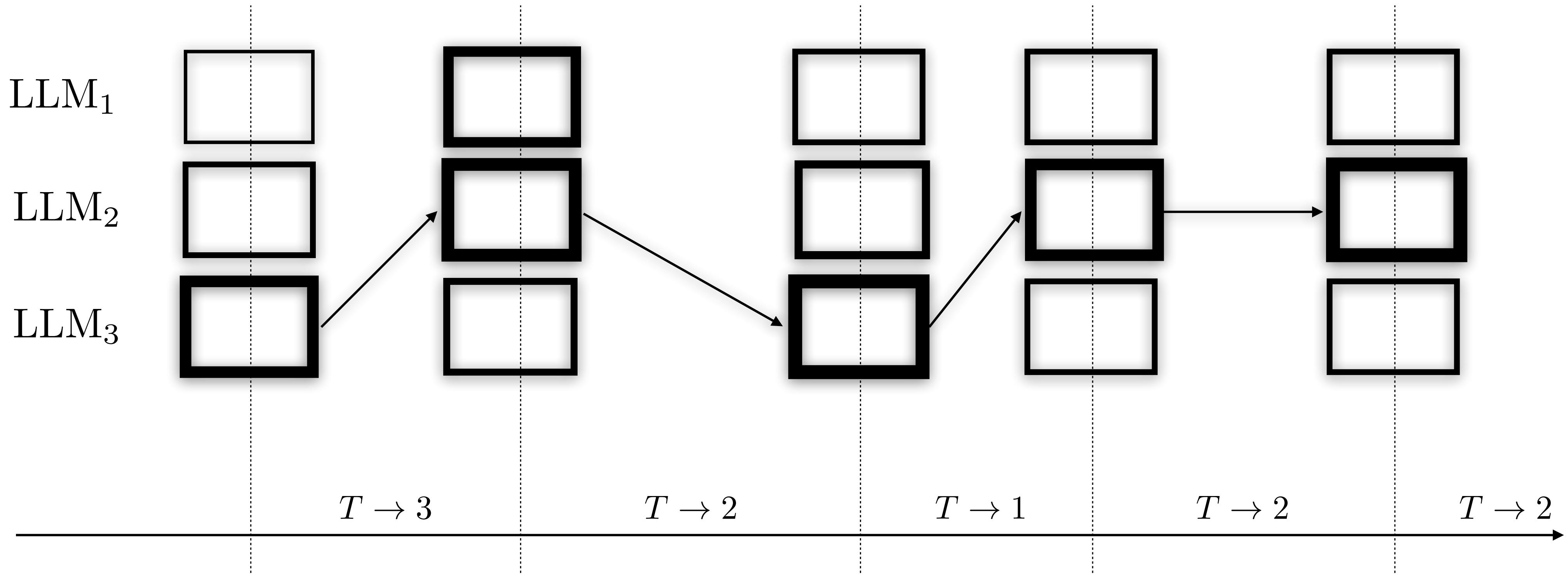
FM

Foundation Models: Intelligence on the cloud

Collection-based Machine Intelligence

S3P-2024 - Collectionless AI

LLMs: LINGUISTIC ENVIRONMENT



THE MAIN QUESTION

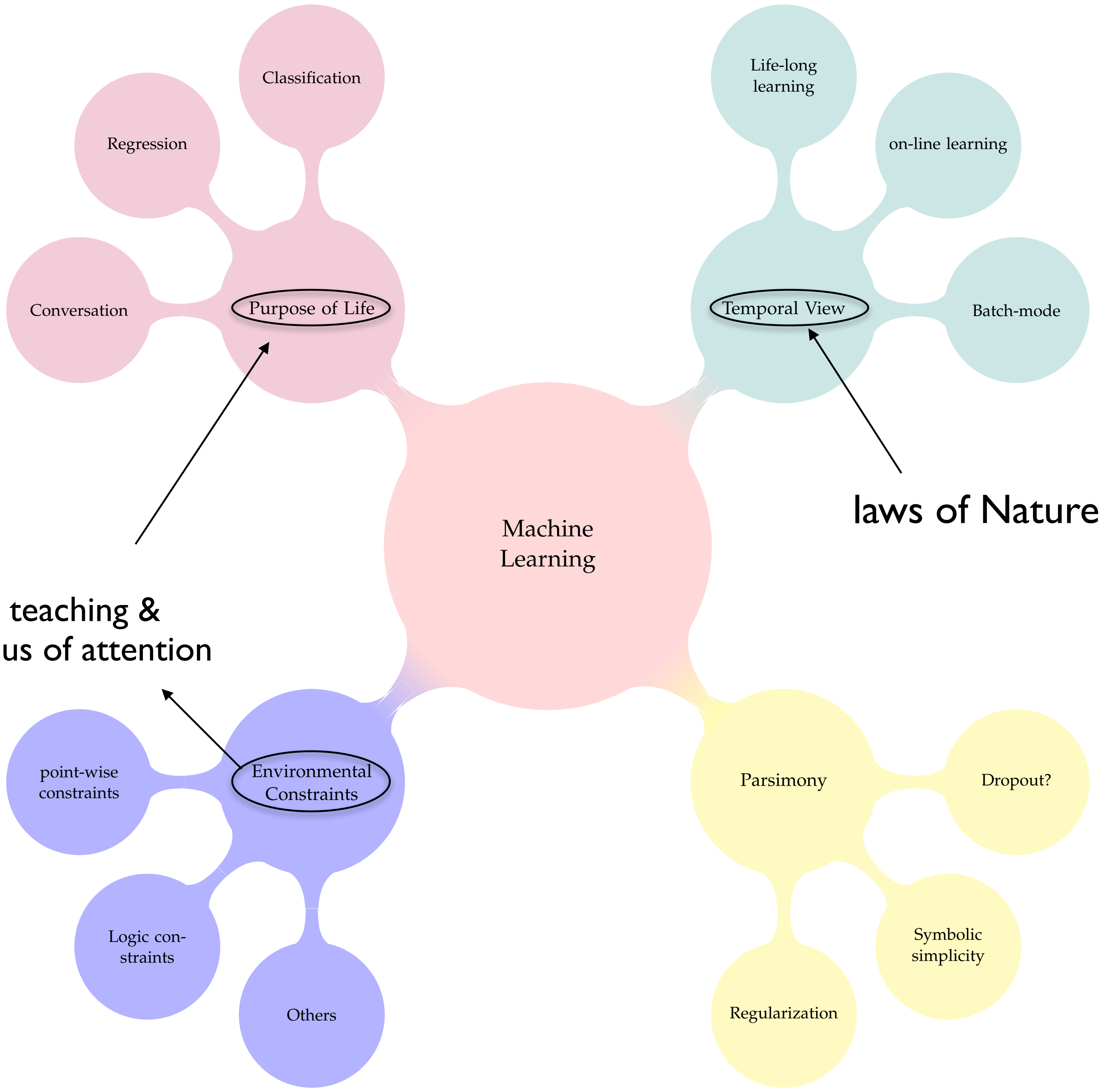
Far better an approximate answer to the right question, which is often vague, than the exact answer to the wrong question, which can always be made precise.

– John Tukey

Learning takes place without accumulation of collections in Nature. Is it possible to devise collectionless machines to gain intelligent skills like in nature?

TOWARDS COLLECTIONLESS AI

computational laws independent of biology



laws of Nature

teaching &
focus of attention

THE PRE-ALGORITHMIC LEVEL

LAWS OF NATURE

ALGORITHMS

big data and huge computational resources
vs intelligent agents that continuously learn in real-world

role of time (time of physics!)

Quality of intelligence:“green deep learning” … focus on efficiency

CRUCIAL ROLE OF TEACHING

DEVELOPMENTAL PSYCHOLOGY

nobody helps your digestion!

collection-based ML

mostly missing
in collection-based!

TIME IS NOT JUST AN ITERATION INDEX!

On-line learning algorithms typically regards time as an iteration index in the function optimization (with the dream of batch-mode)

In human cognition, time is interwound with focussing of attention and active actionss

focusing of attention and teaching involves computational issues, regardless of the “body”!

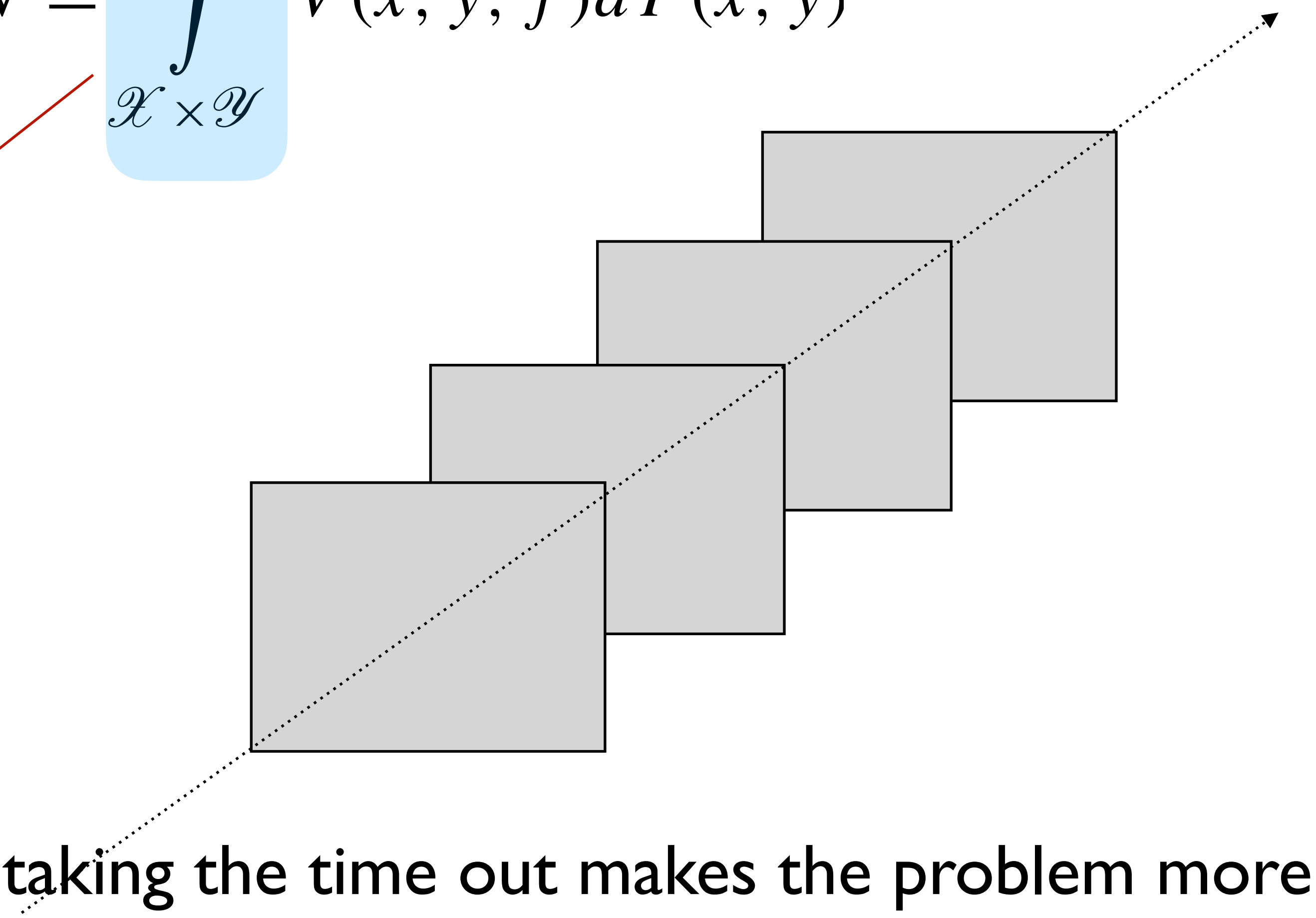
Time

the new Protagonist of Machine Learning

FUNCTIONAL RISK OVER TIME

$$E(f) = \mathrm{E}_{\mathrm{X,Y}}\,\mathrm{V} = \int_{\mathscr{X}\times\mathscr{Y}} V(x, y, f)\,dP(x, y)$$

$$\int_0^T$$



… taking the time out makes the problem more difficult!
Nowadays "artificial static formulations" lead to problems more difficult than what we directly find in nature

FROM LOSS TO LAGRANGIAN

objective functions

$$E(f) = \mathrm{E}_{\mathrm{X,Y}}\,\mathrm{V} = \int_{\mathscr{X}\times\mathscr{Y}} V(x, y, f)\,dP(x, y)$$

approximation over "big data collections"

$$J_{[0,T]} = J_T + \int_0^T dt\ L(x(t), w(t), u(t))$$

approximation over the "lifespan"

value function

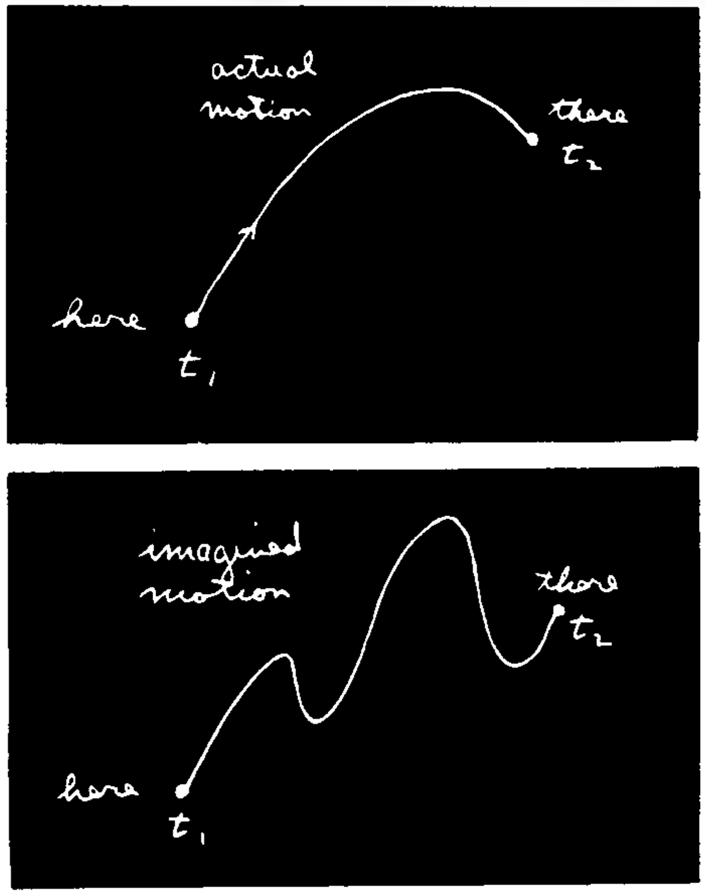
$$V(t, x) = \int_t^T dt\ L(x(t), w(t), u(t))$$

cost to go

S3P-2024 - Intelligence and laws of Nature

TIME IS THE PROTAGONIST OF PHYSICS
Could it be in learning, too?

From Feynman Lectures



Least Action in Mechanics

$$S = \int \left[\frac{m}{2} \left(\frac{dx}{dt} \right)^2 - V(x) \right] dt$$

… thinking of
Least Action in Cognition:
a pre-algorithmic view of (machine) learning

Stationarity of Action

variation

$$S = \int_{t_1}^{t_2} \left[\frac{m}{2} \left(\frac{d\underline{x}}{dt} + \frac{d\eta}{dt} \right)^2 - V(\underline{x} + \eta) \right] dt$$

$$\left(\frac{d\underline{x}}{dt}\right)^2 + 2\,\frac{d\underline{x}}{dt}\,\frac{d\eta}{dt} + \cancel{\left(\frac{d\eta}{dt}\right)^2}$$

$$S = \int_{t_1}^{t_2} \left[\frac{m}{2} \left(\frac{d\underline{x}}{dt} \right)^2 - V(\underline{x}) + m\,\frac{d\underline{x}}{dt}\,\frac{d\eta}{dt} - \eta V'(\underline{x}) + \cancel{\text{(second and higher order)}} \right] dt$$

$$\delta S = m\,\cancel{\frac{d\underline{x}}{dt}\eta(t)}\,\Bigg|_{t_1}^{t_2} - \int_{t_1}^{t_2} \frac{d}{dt}\left(m\,\frac{d\underline{x}}{dt} \right) \eta(t)\, dt - \int_{t_1}^{t_2} V'(\underline{x})\, \eta(t)\, dt$$

known boundaries … or transversality conditions

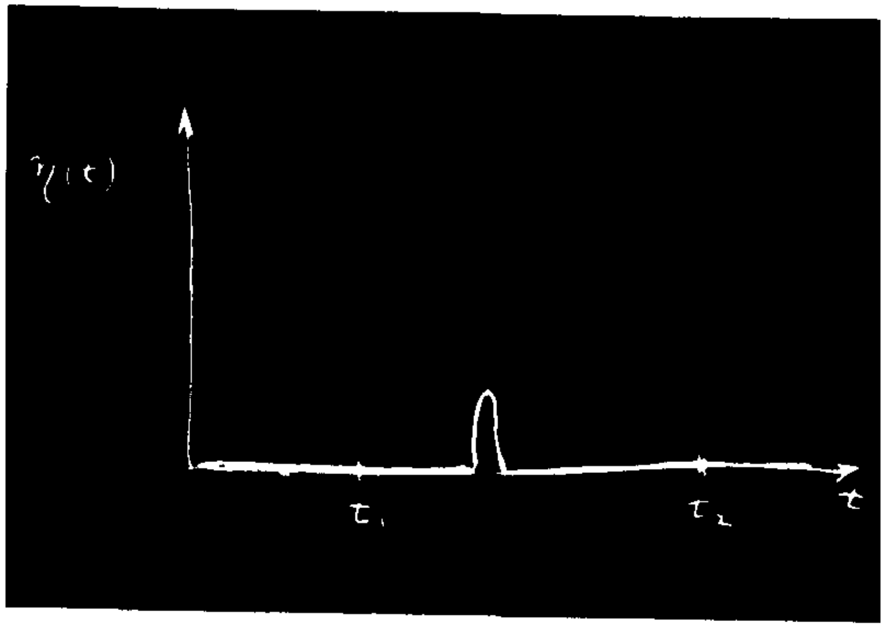
$$\delta S = \int_{t_1}^{t_2} \left[-m\,\frac{d^2\underline{x}}{dt^2} - V'(\underline{x}) \right] \eta(t)\, dt$$

Stationarity of Action (con’t)

$$\delta S = \int_{t_1}^{t_2} \left[-m \frac{d^2 \underline{x}}{dt^2} - V'(\underline{x}) \right] \eta(t)\, dt.$$

$$\int F(t)\, \eta(t)\, dt = 0.$$

Fundamental Lemma of variational calculus



$$\left[-m \frac{d^2 \underline{x}}{dt^2} - V'(\underline{x}) \right] = 0$$

Euler-Lagrange equations

NATURAL LAWS OF COGNITION: A Pre-Algorithmic Step

Natural Learning Theory $\leadsto$ Mechanics	Remarks
$w_i \leadsto q_i$	Weights are interpreted as generalized coordinates.
$\dot{w}_i \leadsto \dot{q}_i$	Weights variations are interpreted as generalized velocities.
$\upsilon_i \leadsto p_i$	The conjugate momentum to the weights is defined by using the machinery of Legendre transforms.
$A(w) \leadsto S(q)$	The cognitive action is the dual of the action in mechanics.
$F(t, w, \dot{w}) \leadsto L(t, q, \dot{q})$	The Lagrangian F is associated with the classic Lagrangian L in mechanics.
$H(t, w, \upsilon) \leadsto H(t, q, p)$	When using w and υ, we can define the Hamiltonian, just like in mechanics.

weights of neural net as
position particles

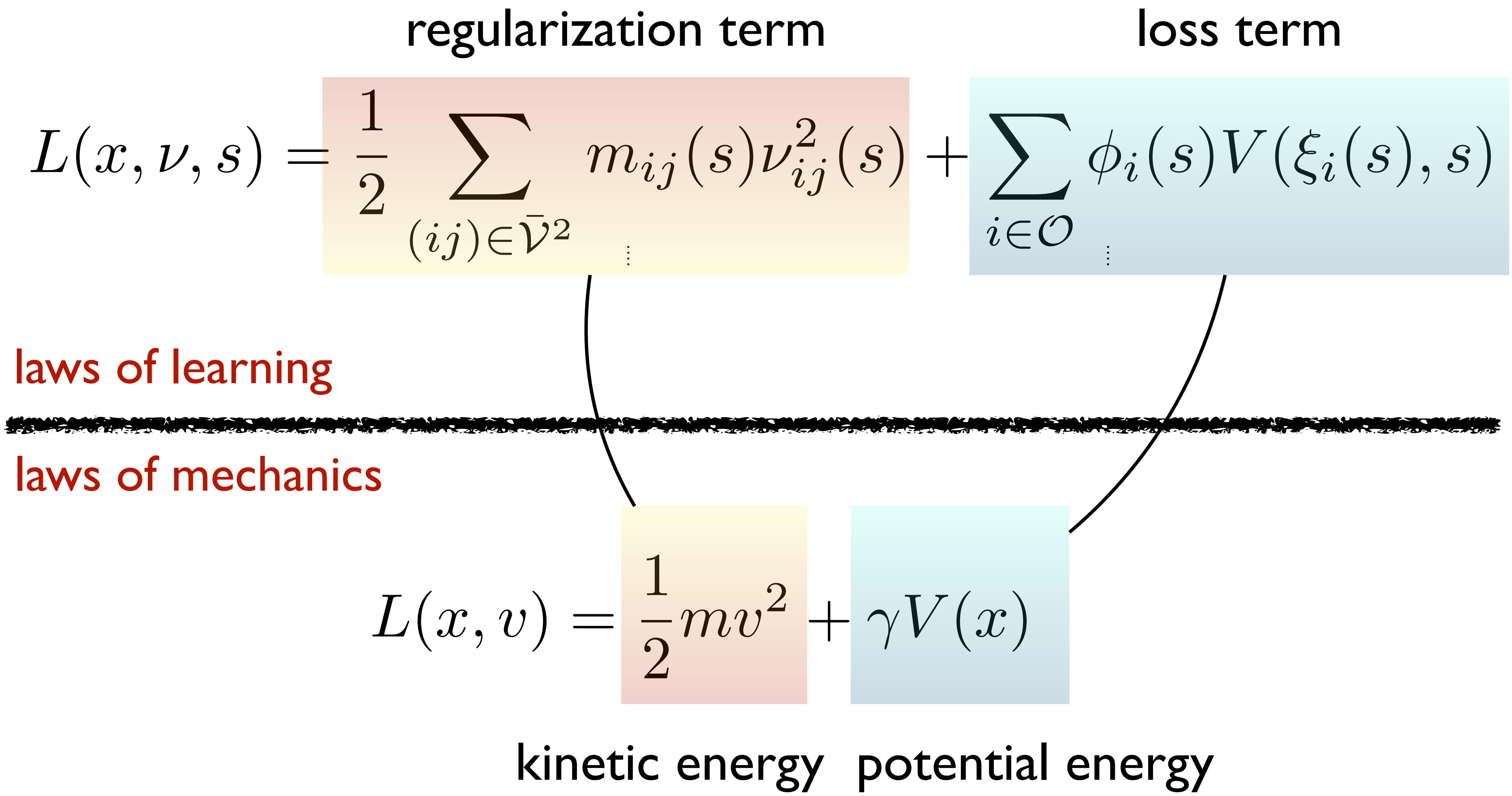
Learning and Optimality

Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. (See Bellman, 1957, Chap. III.3.)

Principle of Causal Optimality: An optimal causal policy has the property that, at a given time, the past decisions cannot be changed. The decisions are based on the current state and the information available at that time.

LINKS WITH MECHANICS

Causal Optimization and Dissipation



regularization term

loss term

$$L(x,\nu,s) = \underbrace{\frac{1}{2}\sum_{(ij)\in\bar{\mathcal{V}}^2} m_{ij}(s)\nu_{ij}^2(s)}_{} + \underbrace{\sum_{i\in\mathcal{O}}\phi_i(s)V(\xi_i(s),s)}_{}$$

laws of learning

laws of mechanics

$$L(x,v) = \frac{1}{2}mv^2 + \gamma V(x)$$

kinetic energy

potential energy