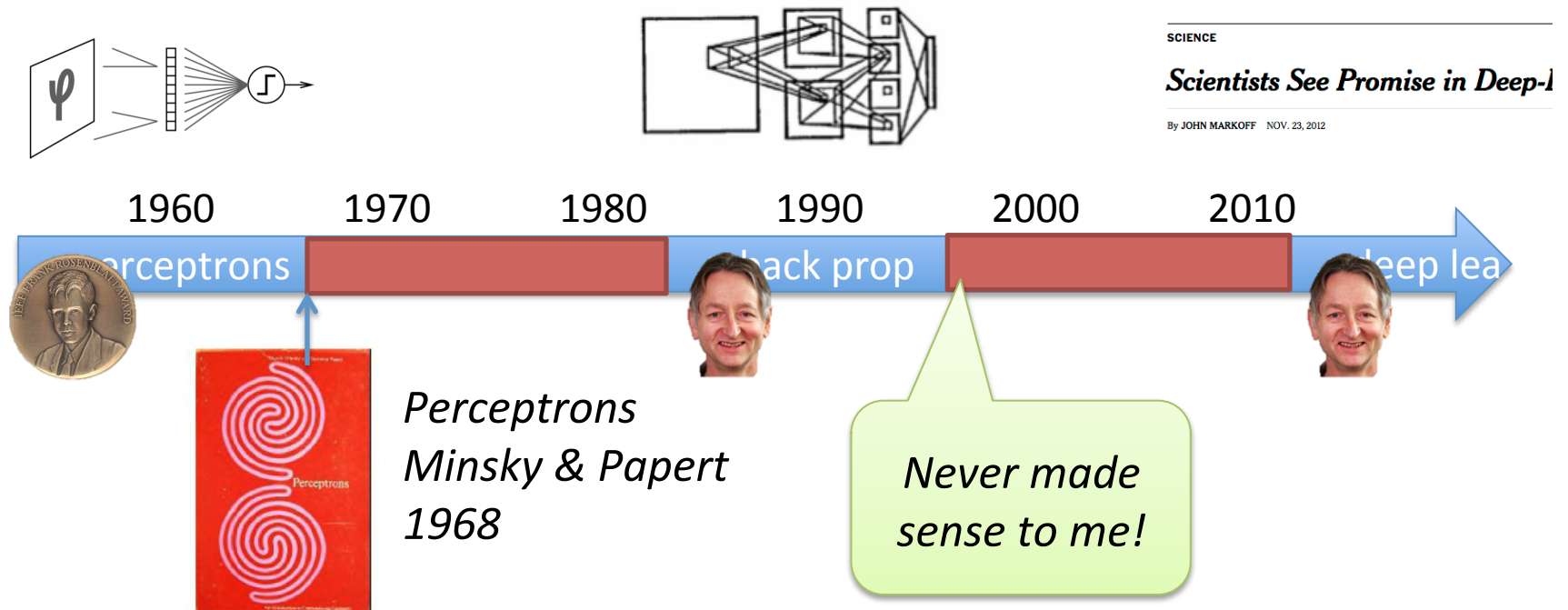


“Perceptrons” revisited

Léon Bottou

Facebook AI Research

Good ideas do not die?



Extinctions do not happen without real reasons.
Time to re-read “Perceptrons”.

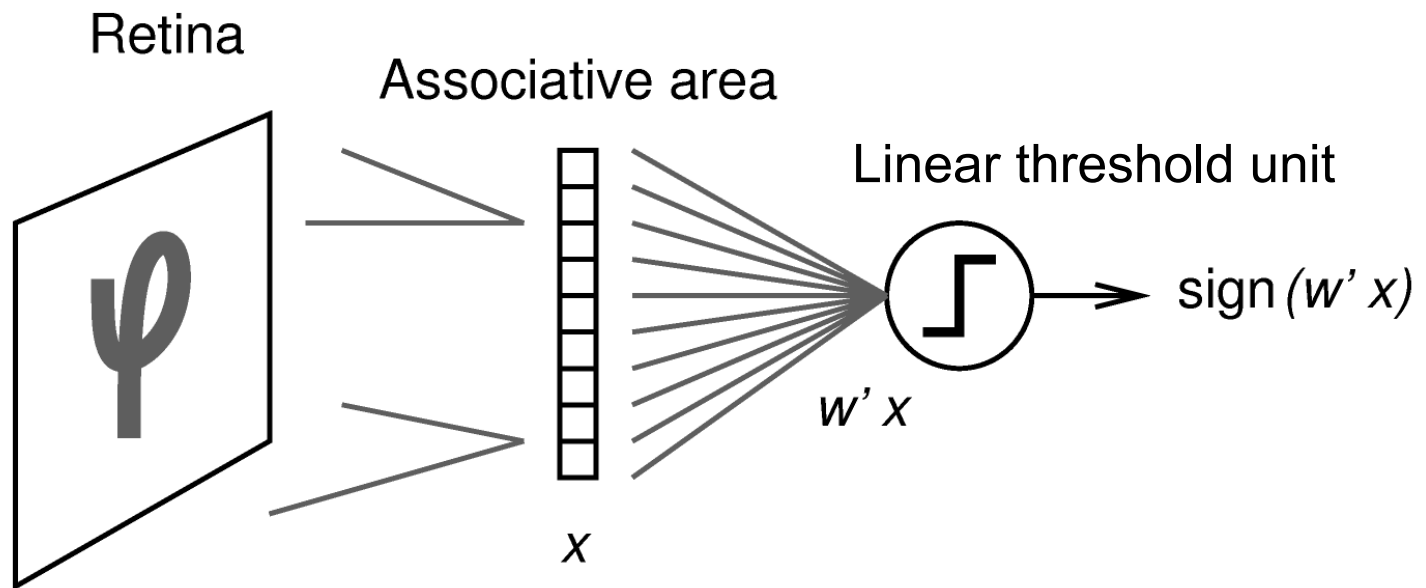
A Random Walk...

1. What's in the book?
2. Computer Science versus Cybernetics
3. Building on the Work of Others

The perceptron algorithm

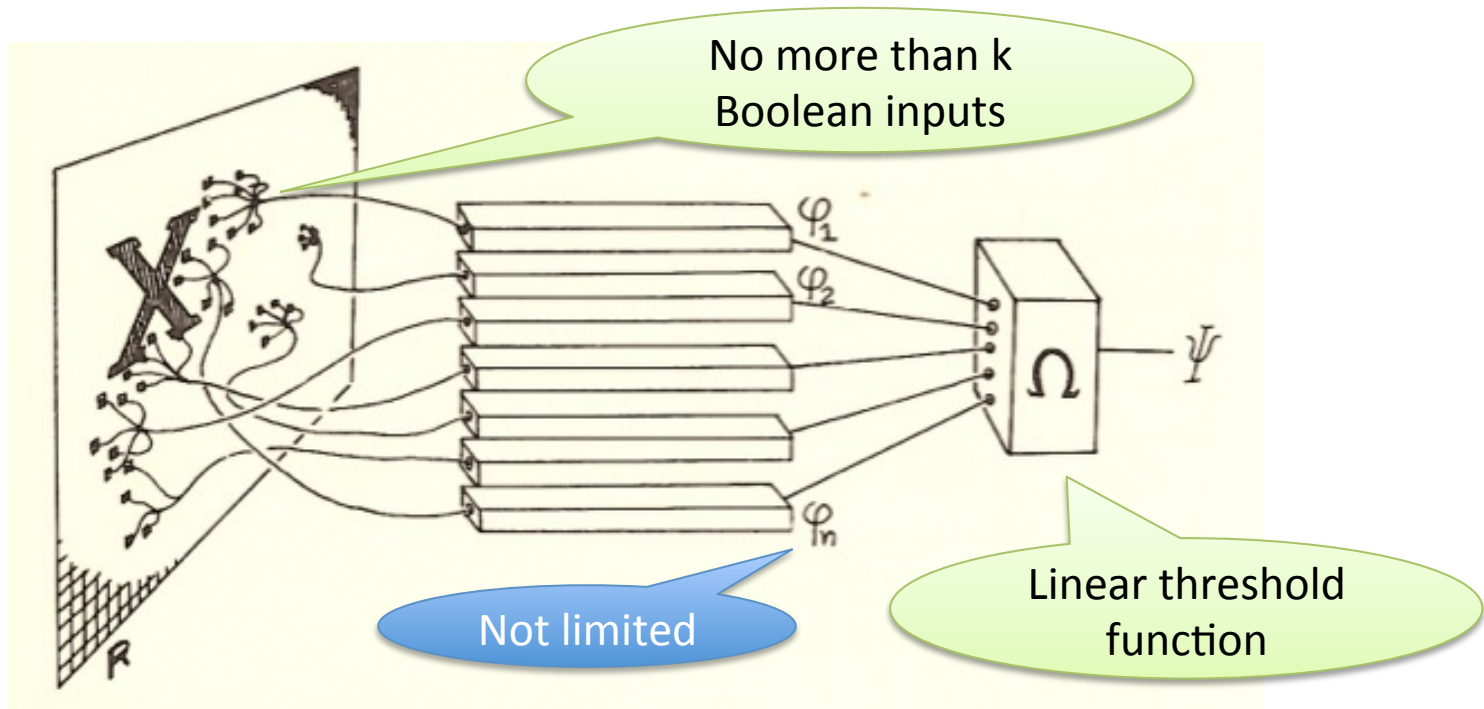
I. What's in the book?

The perceptron



Supervised learning of using the **perceptron algorithm**.

Order-k Perceptron




Metaphor for parallel computation

- Targets what can be computed by Rosenblatt's perceptron.
- Also describes what can be computed by a convnet with a final pooling layer.
- Similar techniques could characterize what can be computed with map/reduce

Boolean predicates and Perceptrons

Take simple Boolean predicates and establish their order requirements.

- Focus on group invariant predicates:
 - Parity has “infinite” order.
- Geometrical predicates:
 - Connectedness has infinite order.
 - Euler number has low order.
 - Etc. with caveats



*This is
absolutely
brilliant!*

Strong opinions

13.5 Why Prove Theorems?

Why did you prove all these complicated theorems? Couldn't you just take a perceptron and see if it can recognize $\psi_{\text{CONNECTED}}$?

No.

Strong opinions

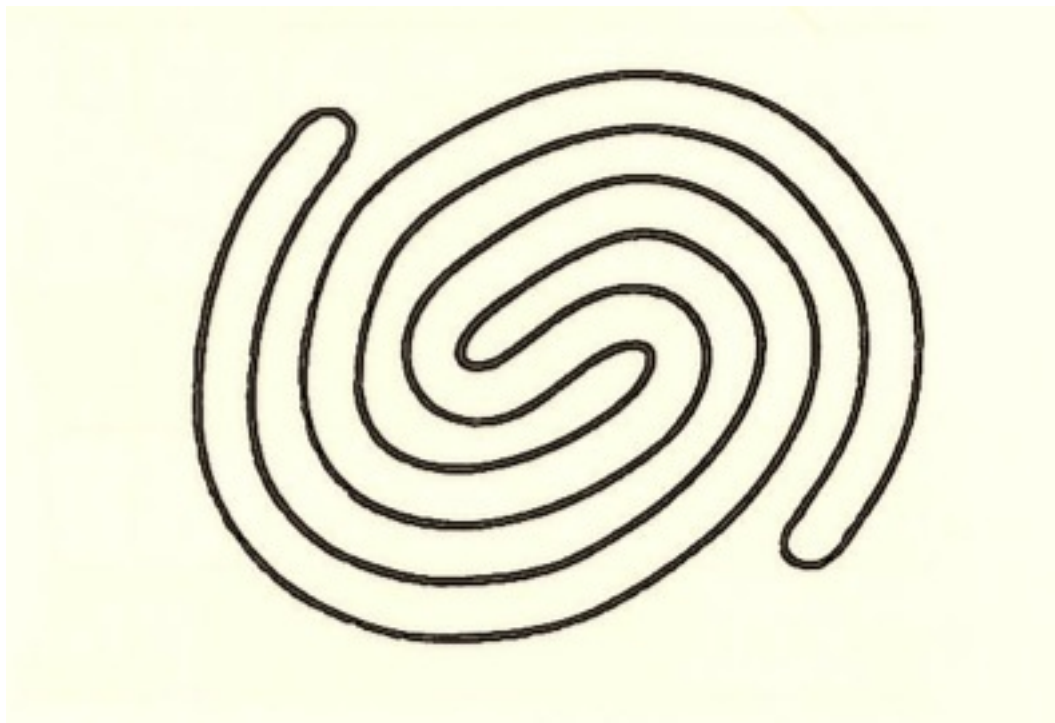
0.3 Cybernetics and Romanticism

The machines we will study are abstract versions of a class of devices known under various names; we have agreed to use the name "perceptron" in recognition of the pioneer work of Frank Rosenblatt. Perceptrons make decisions—determine whether or

Our discussion will include some rather sharp criticisms of earlier work in this area. Perceptrons have been widely publicized as "pattern recognition" or "learning" machines and as such have been discussed in a large number of books, journal articles, and voluminous "reports." Most of this writing (some exceptions are mentioned in our bibliography) is without scientific value and we will not usually refer by name to the works we criticize. The sciences of computation and cybernetics began,

Connectedness

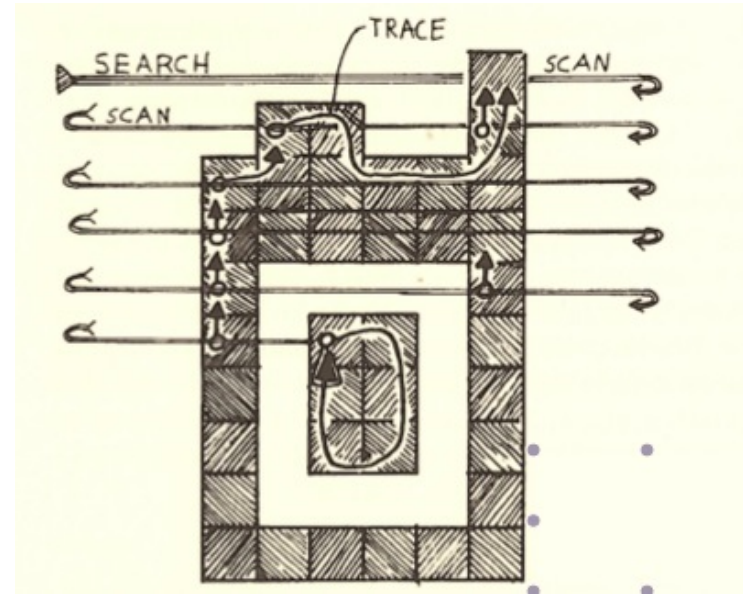
Is a shape made of a single connected component?



No small-order perceptron can say.

Connectedness

But a simple algorithm provably can compute “connectedness”



Theorem 9.2: For any ϵ there is a 2-symbol Turing machine that can verify the connectedness of a figure X on any rectangular array R , using less than $(2 + \epsilon) \log_2 |R|$ squares of tape.

Is connectedness important?

13.3 Analyzing Real-World Scenes

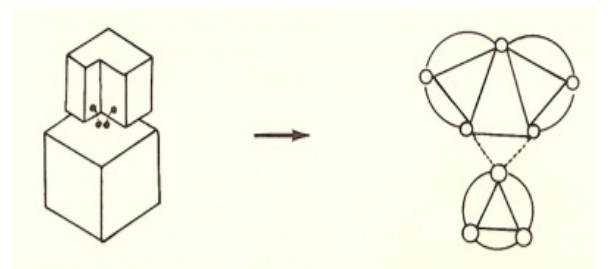
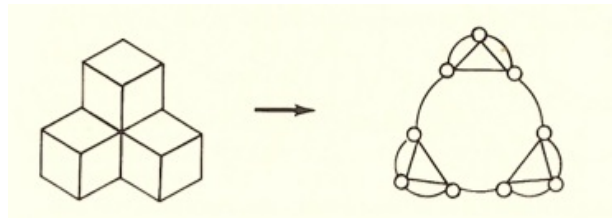
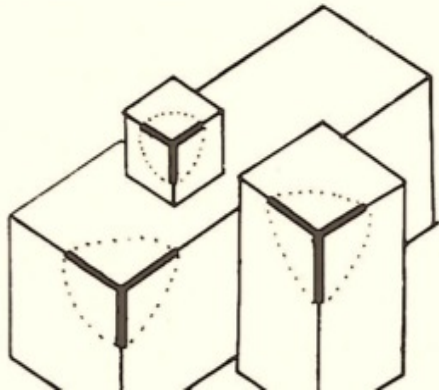
One can understand why you, as mathematicians, would be interested in such clear and simple predicates as ψ_{PARITY} and $\psi_{\text{CONNECTED}}$. But what if one wants to build machines to recognize chairs and tables or people? Do your abstract predicates have any relevance to such problems, and does the theory of the simple perceptron have any relevance to the more complex machines one would use in practice?

This is a little like asking whether the theory of linear circuits has relevance to the design of television sets. Absolutely, some concept of connectedness is required for analyzing a scene with many objects in it. For the whole is just the sum of its parts and

Scene analysis

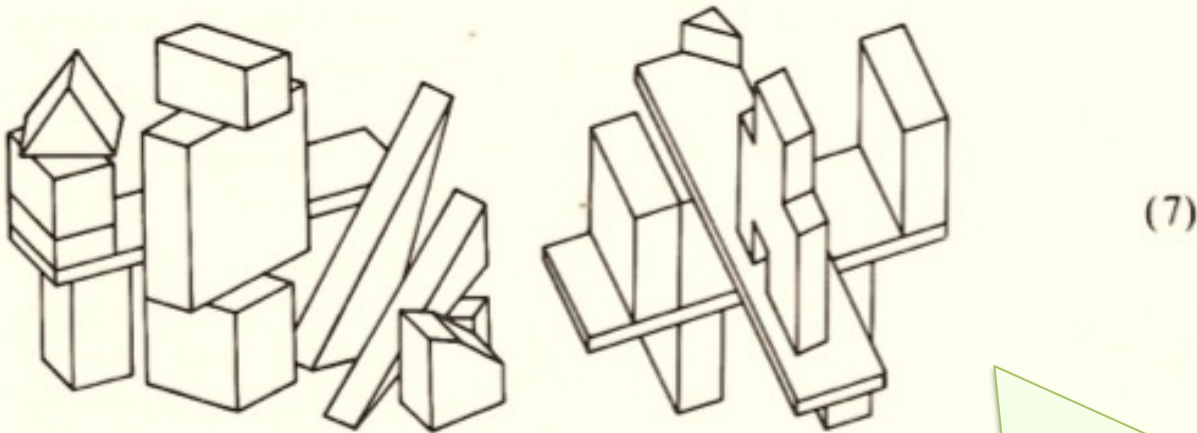
13.4 Guzman's Approach to Scene-Analysis

In scenes like this,



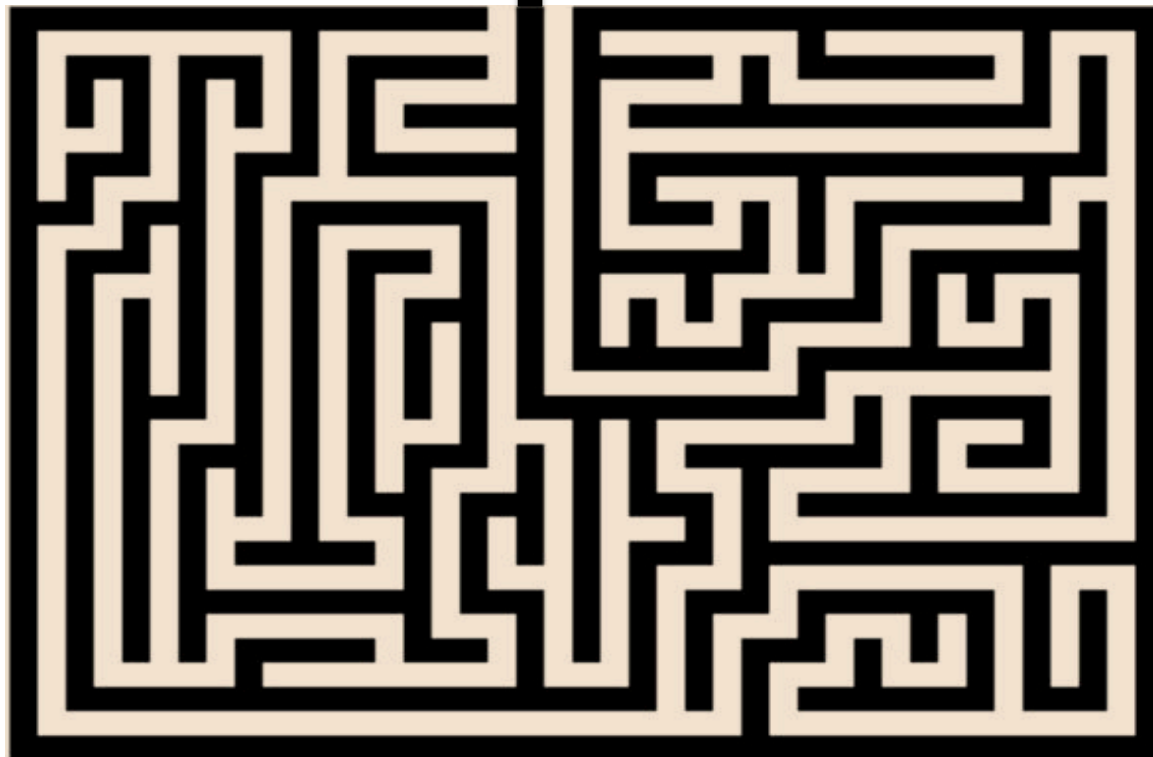
Scene analysis

and the effects of some vertices are modified by their associations with others. This variety of resources enables the program to solve scenes like



- A smart program may provably solve such “dreamed” scenes.
- But why should scenes like this be relevant to real scenes.
- *Are they letting us down?*

Is connectedness easy for us?



What is easy for us?



This shape
represents a
mouse

This shape
represents a
piece of cheese

Are there provable algorithms for



“Mouseness?”



“Cheesiness?”

- “Connectedness” has a clear and compact mathematical specification.
- “Mouseness” and “cheesiness” do not.

Why prove theorems?

13.5 Why Prove Theorems?

Why did you prove all these complicated theorems? Couldn't you just take a perceptron and see if it can recognize $\psi_{\text{CONNECTED}}$?

No.

Are we dealing with
an abstract science (like maths) or
an empirical science (like physics)?

In the case of an empirical science,

- we cannot only prove theorems, and
- we cannot only run experiments either.

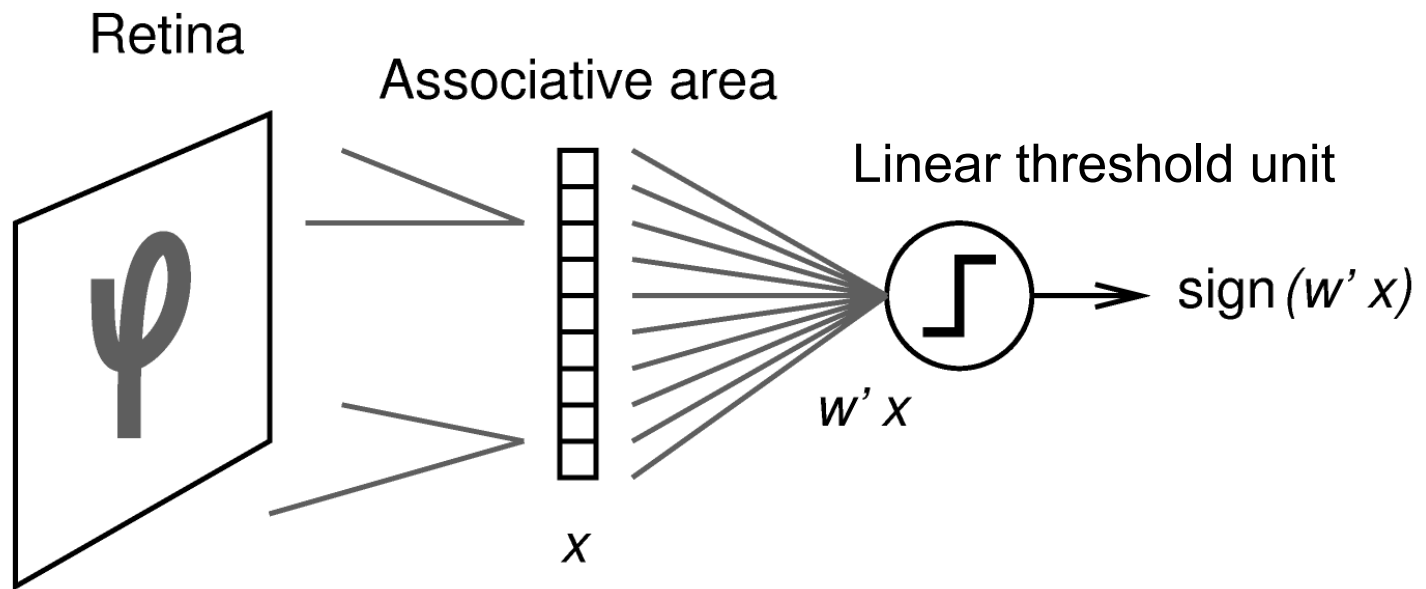


Their answer
is technically
correct

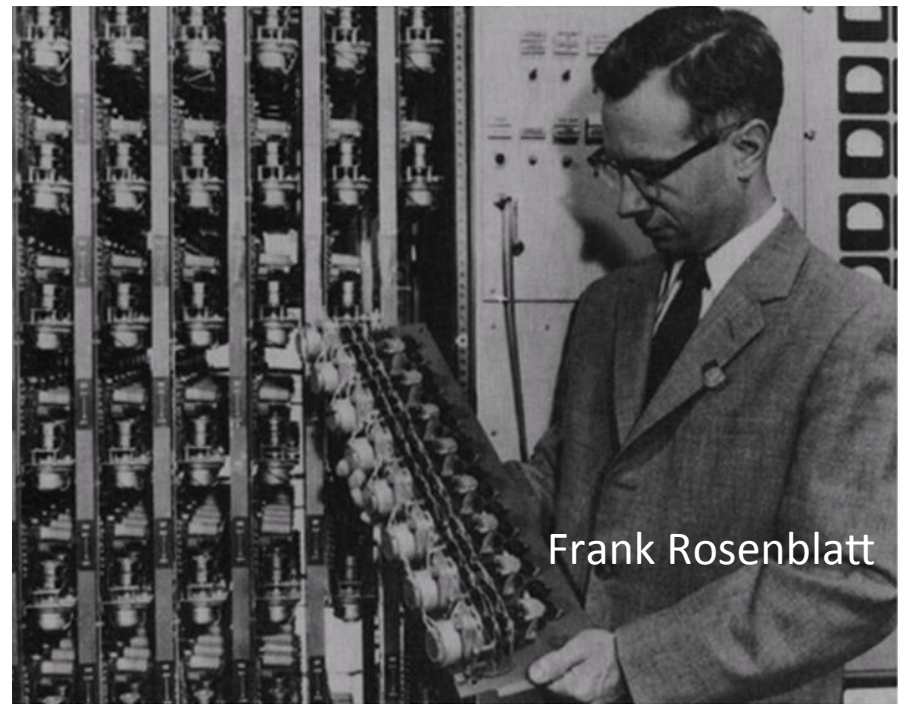


II. Computer Science versus Cybernetics

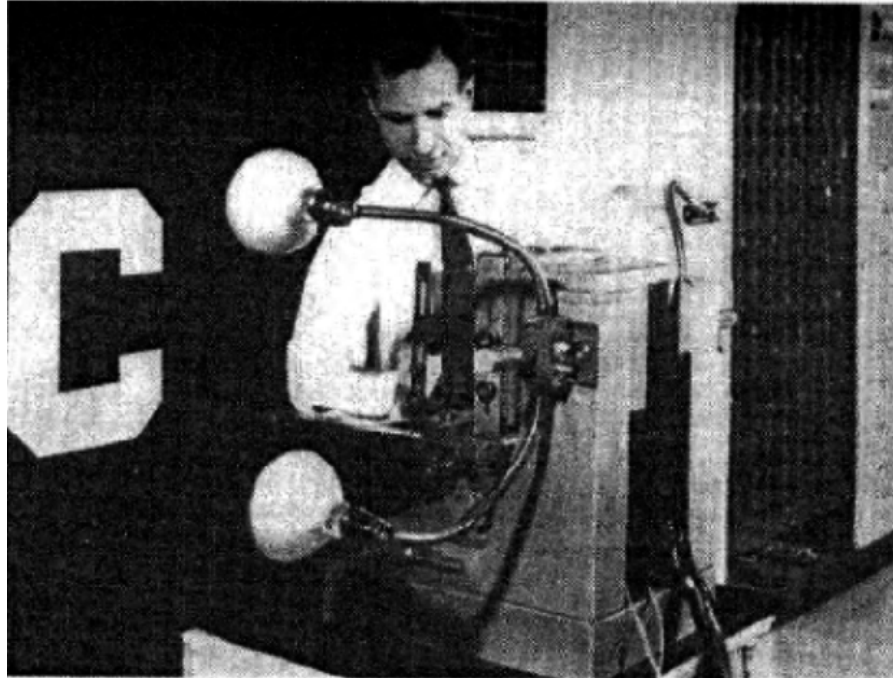
The perceptron is not
an algorithm that runs on a computer



The perceptron is a machine



The perceptron is the computer



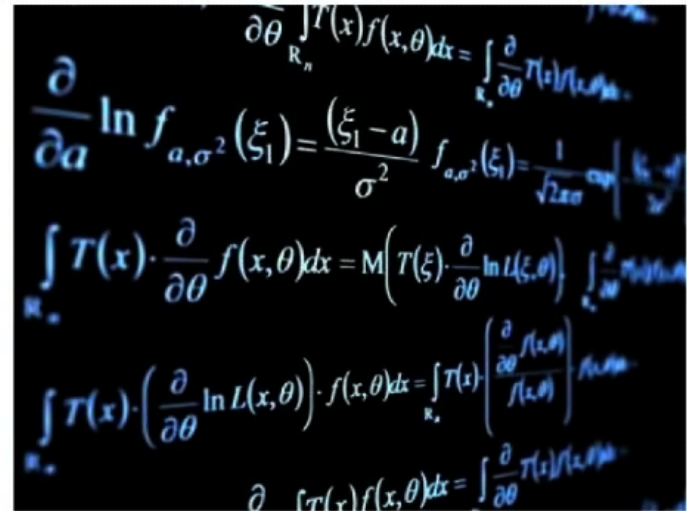
- Tasks that vintage computers could not match.
- Alternative computer architecture? Analog computer?

How to build computing machines?

Biological computer



Mathematical computer

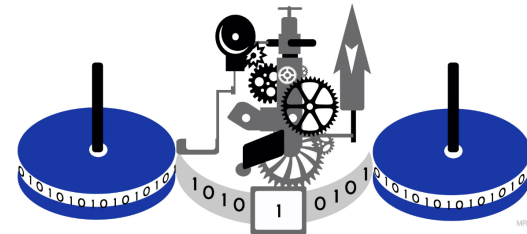


- Which model to emulate : brain or mathematical logic ?
- **Mathematical logic has won. Why?**

Computing with symbols

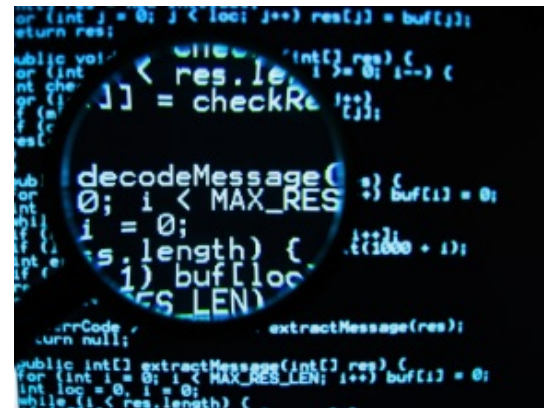
General computing machines

- Turing machine
- von Neumann machine



Engineering

- Programming
= reducing a complex task into a collection of simple tasks.
- Computer language
- Debugging
- OS
- API



CS as an Abstract Science

What constitutes a result?

- ✓ An algorithm that
- ✓ provably fulfills a mathematical specification
- ✓ with bounds on its resource demands.

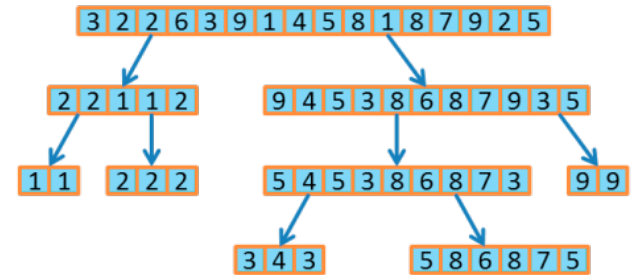
No need for a real computer.

No need for data either.

CS as an Abstract Science

Example:

- ✓ Quicksort
- ✓ sorts an array of n elements
- ✓ in $O(n \log n)$ time and $O(n)$ memory



Such elementary results are fantastically useful.

Programming = Reducing a complex problem to a collection of simpler problems amenable to known algorithms.

Analogy with proving mathematical theorems

Building on the work of others



$$\begin{aligned} \frac{\partial}{\partial a} \ln f_{a,\sigma^2}(\xi_i) &= \frac{(\xi_i - a)}{\sigma^2} f_{a,\sigma^2}(\xi_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(\xi_i - a)^2}{2\sigma^2}\right\} \frac{(\xi_i - a)}{\sigma^2} \\ \int \tau(x) \frac{\partial}{\partial \theta} f(x, \theta) dx &= M\left(\tau(\xi) \frac{\partial}{\partial \theta} \ln l(\xi, \theta)\right) \\ \int \tau(x) \left(\frac{\partial}{\partial \theta} \ln l(x, \theta)\right) f(x, \theta) dx &= \int \tau(x) \left(\frac{\partial}{\partial \theta} f(x, \theta)\right) dx \\ \frac{\partial}{\partial \theta} \int \tau(x) f(x, \theta) dx &= \int \tau(x) \left(\frac{\partial}{\partial \theta} f(x, \theta)\right) dx \end{aligned}$$

How to build computers?

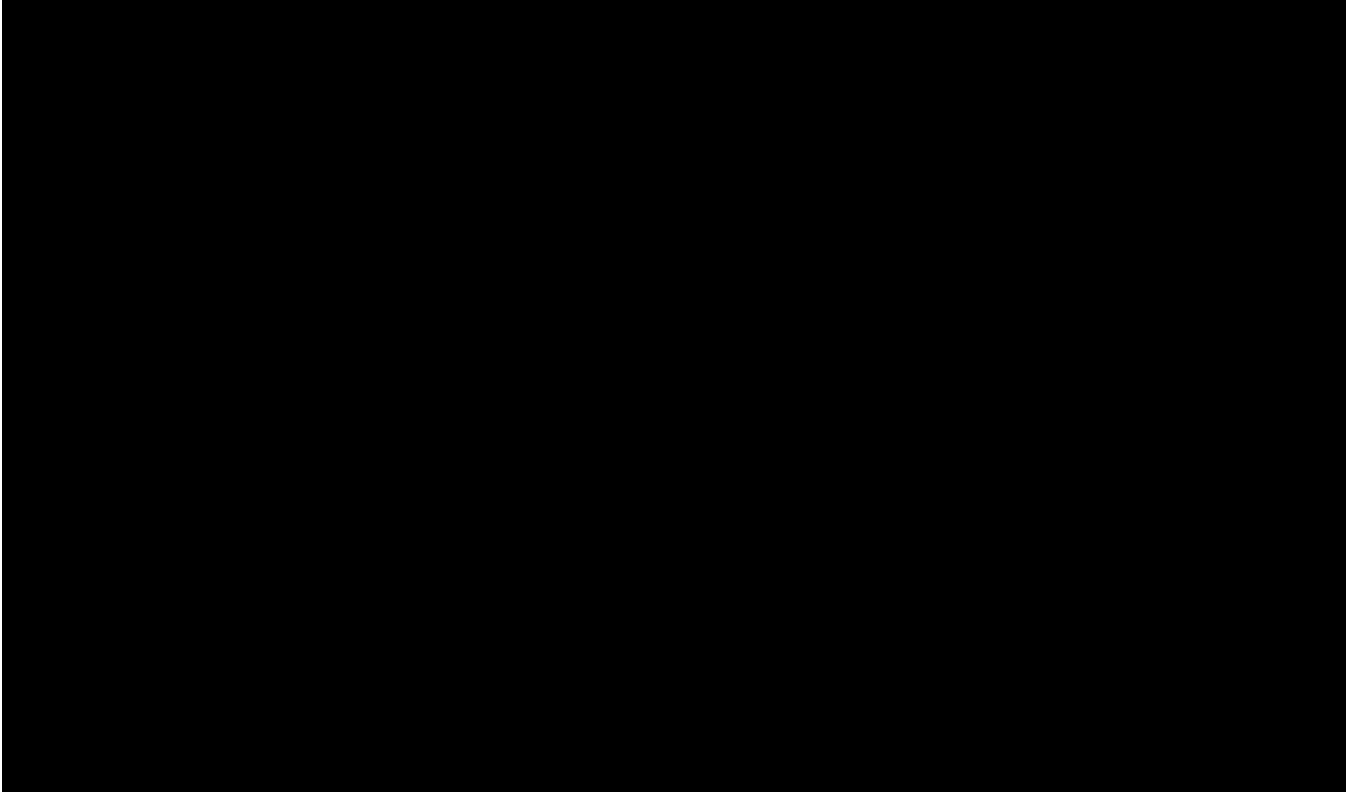
Mathematical logic has won.

Why?

- Big things demand lots of people.
- Getting lots of people to work together is hard.
- Mathematical logic offers a lot of support for good computer science engineering practices.

Evidence: a significant fraction of the CS literature aims at helping collaboration (code reuse, software components,...)

Same place (MIT)
About the same time (1964-68)



The book acknowledges productive input from R. M. Fano.
Watch the full video on YouTube!

Learning as a software components

“ML system X recognizes faces with 95% accuracy”

- This statement only applies to a precise distribution of testing examples.
- Using X with different faces may not work as well.
Example: Using X with children faces may not work at all because the children face are precisely the 5% that did not work well.
- This is painful for programmers.
Using X in an innovative way may not work at all. One needs to rebuild X for the new use. It may or may not work.
- Things get worse if the component keeps learning...

Learning as a software components

Machine Learning: The High-Interest Credit Card of Technical Debt

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young
{dsculley, gholt, dgg, edavydov}@google.com
{toddphillips, ebner, vchaudhary, mwyong}@google.com
Google, Inc

Abstract

2.1 Entanglement

From a high level perspective, a machine learning package is a tool for mixing data sources together. That is, machine learning models are machines for creating entanglement and making the isolation of improvements effectively impossible.

Algorithmic ML Theorems

A beautiful theorem

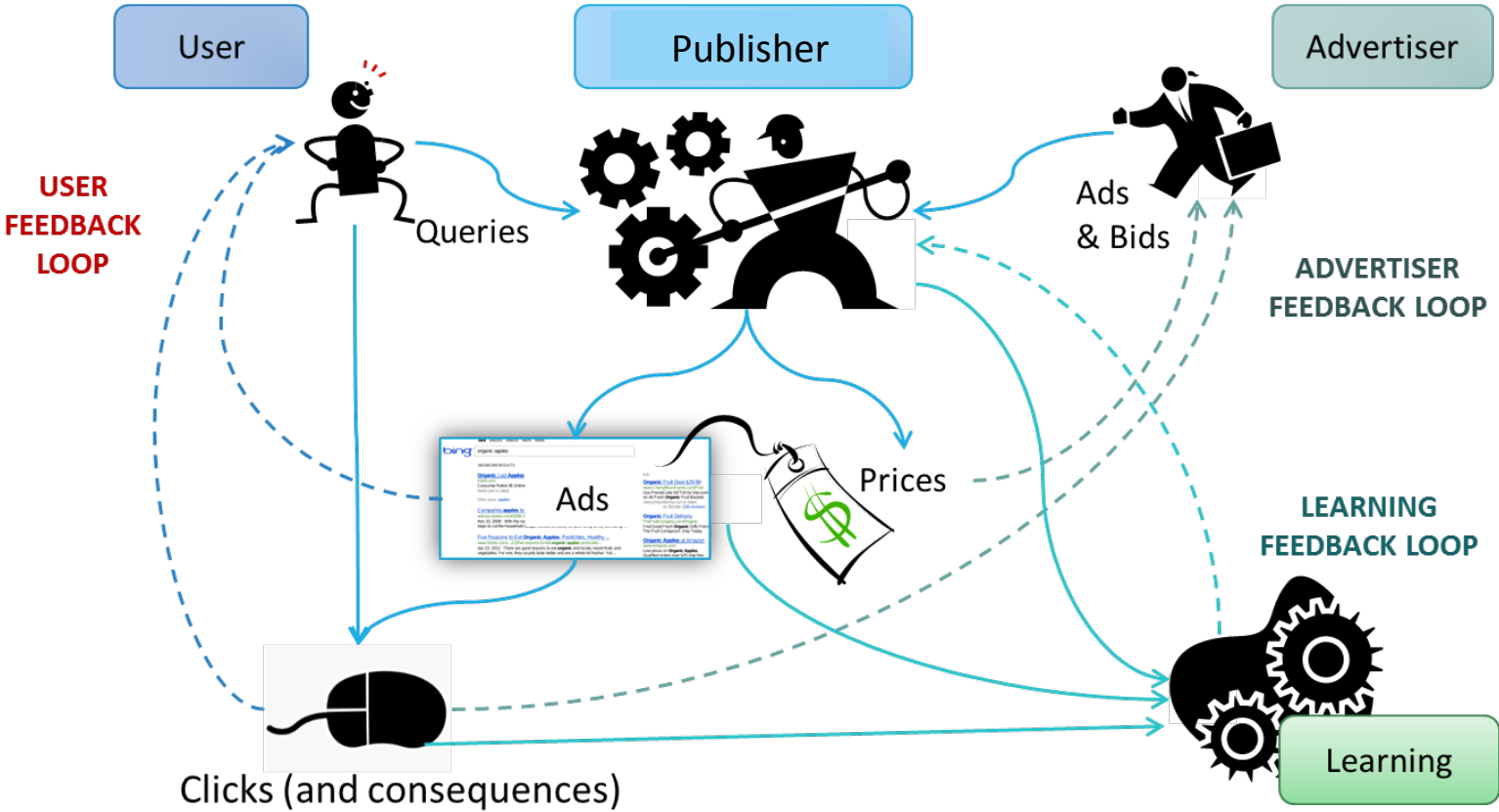
- ✓ UCB solves the
- ✓ multi-armed bandit problem
- ✓ with a sub-linear regret bound



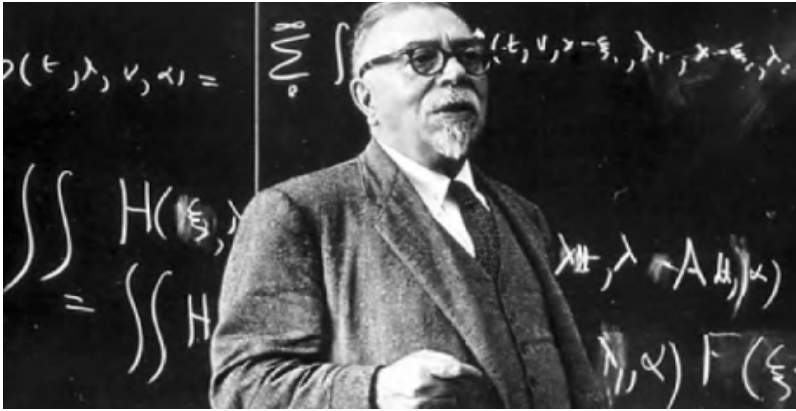
Although the proof contains useful insights,
such a theorem is useless in practice.

*We do not know how to reduce complex ML
problems to a collection of simple ML problems.*

Feedback loops



Studying feedback loops



Another MIT book.

- Norbert Wiener
Cybernetics, 1948.

- Information (signal) feedback loops are everywhere. They are central to adaptation and learning.
- They should be the object of a new scientific discipline, *cybernetics*.
- *Although things did not happen exactly in this way, the previous slide makes a case for this point of view.*

Two caricatures of ML

Machine Learning

Learning is a topic for algorithm research.

- Study learning algorithms and describe their properties
- Data or computers are not absolutely needed to reach the “perfect” learning algorithm.
- Problem specific feature engineering is something engineers do.

Learning Machines

Learning is more fundamental than algorithms.

- Observe how the system transforms input signals and leverages training signals.
- Find ways to make it better.
- Features are transformations of the input signal. One should learn them as well.
- Modeling experimental observations is required.

Two caricatures of ML

Machine Learning

Learning is a
algorithm.

- Choosing algorithms and
data to solve their problems
- Data or algorithms not
able to reach
learning
algorithm.
- Problem specific feature
engineering is something
engineers do.

"COLT" STYLE

"CONVEX ONLY"

Learning Machines

Learn from
the environment

- Observe how the system
transforms inputs and
signals
- Feature engineering is
not needed.
- Feature engineering is
of the input signals should
learn
- More environmental
observations is required.

"NIPS" STYLE

"PERCEPTRON"

"DEEP LEARNING"

"RNNS"

Quiz

Machine Learning

Learning is a general algorithm that can be applied to a wide range of problems.

- General learning algorithms are used to solve their problems.
- Data or examples are used to reach a solution.
- Problem specific feature engineering is something engineers do.

“COLT” STYLE

“CONVEX ONLY”

BAYES NET ?

VAPNIK ?

Learning Machines

Learning machines are designed to solve specific problems.

- Learning machines are designed to solve specific problems.
- Learning machines are designed to solve specific problems.
- Learning machines are designed to solve specific problems.

“NIPS” STYLE

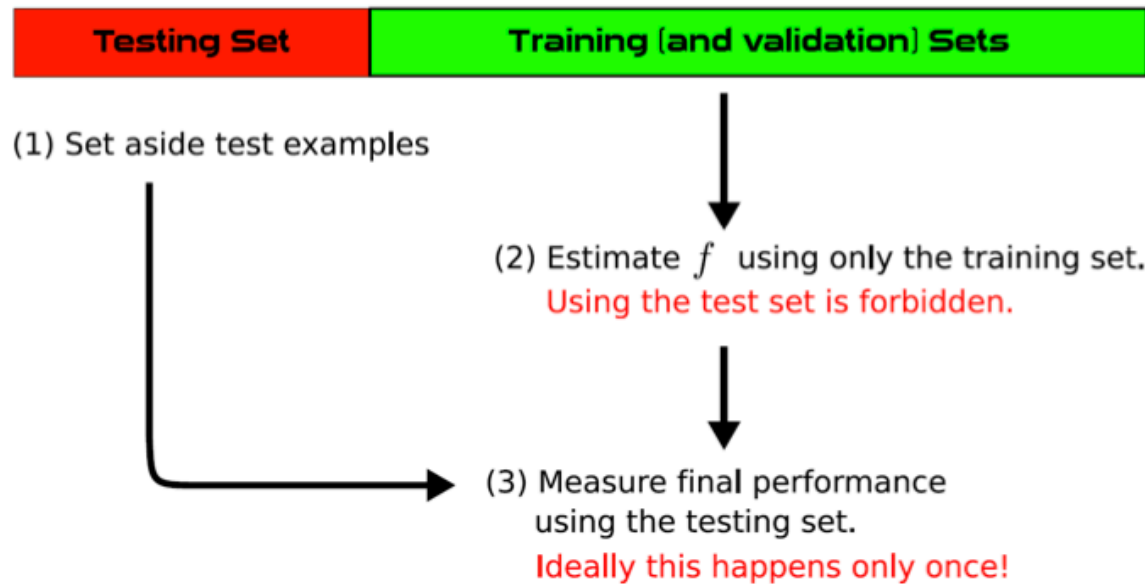
PERCEPTRON

“DEEP LEARNING”

“RECURRENT NETWORKS”

Learning as an Empirical Science

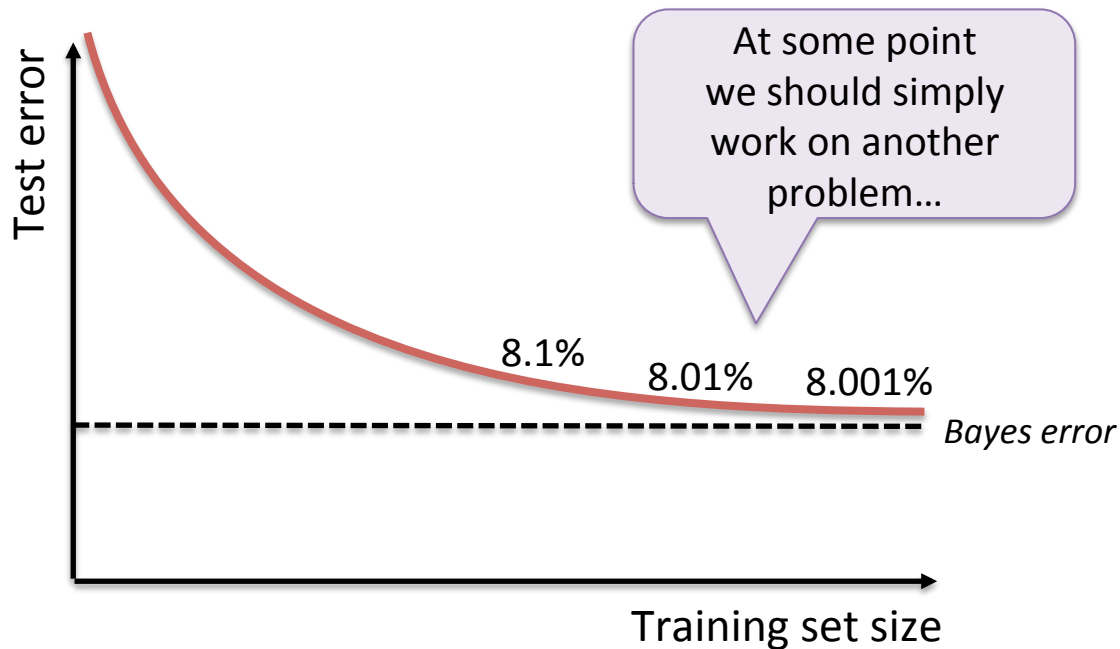
Progress during the last few decades has been driven by a single experimental paradigm.



Learning as an Empirical Science

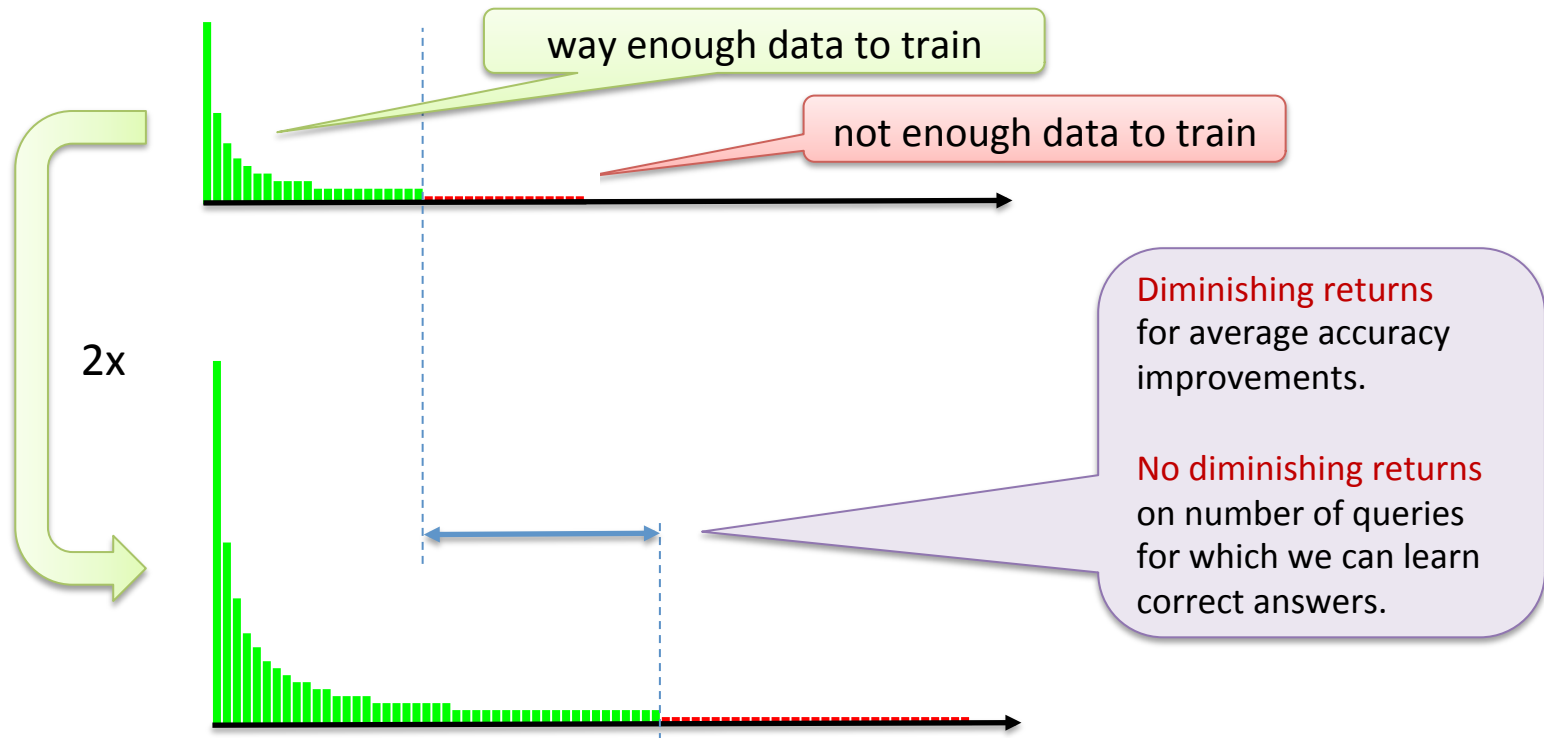
Limitations of our single experimental paradigm

Diminishing returns



Learning as an Empirical Science

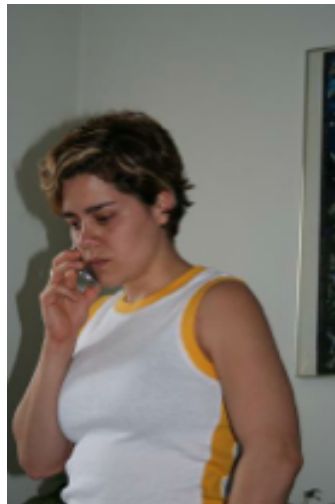
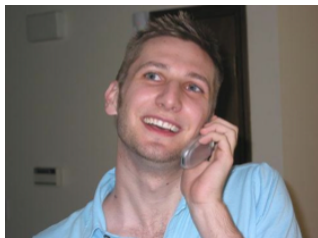
Accuracy vs. coverage



Learning as an Empirical Science

Concepts \neq Statistics

Example: detection the action “phoning”



(Oquab et al., CVPR 2014)

Learning as an Empirical Science

Limitations of our single experimental paradigm

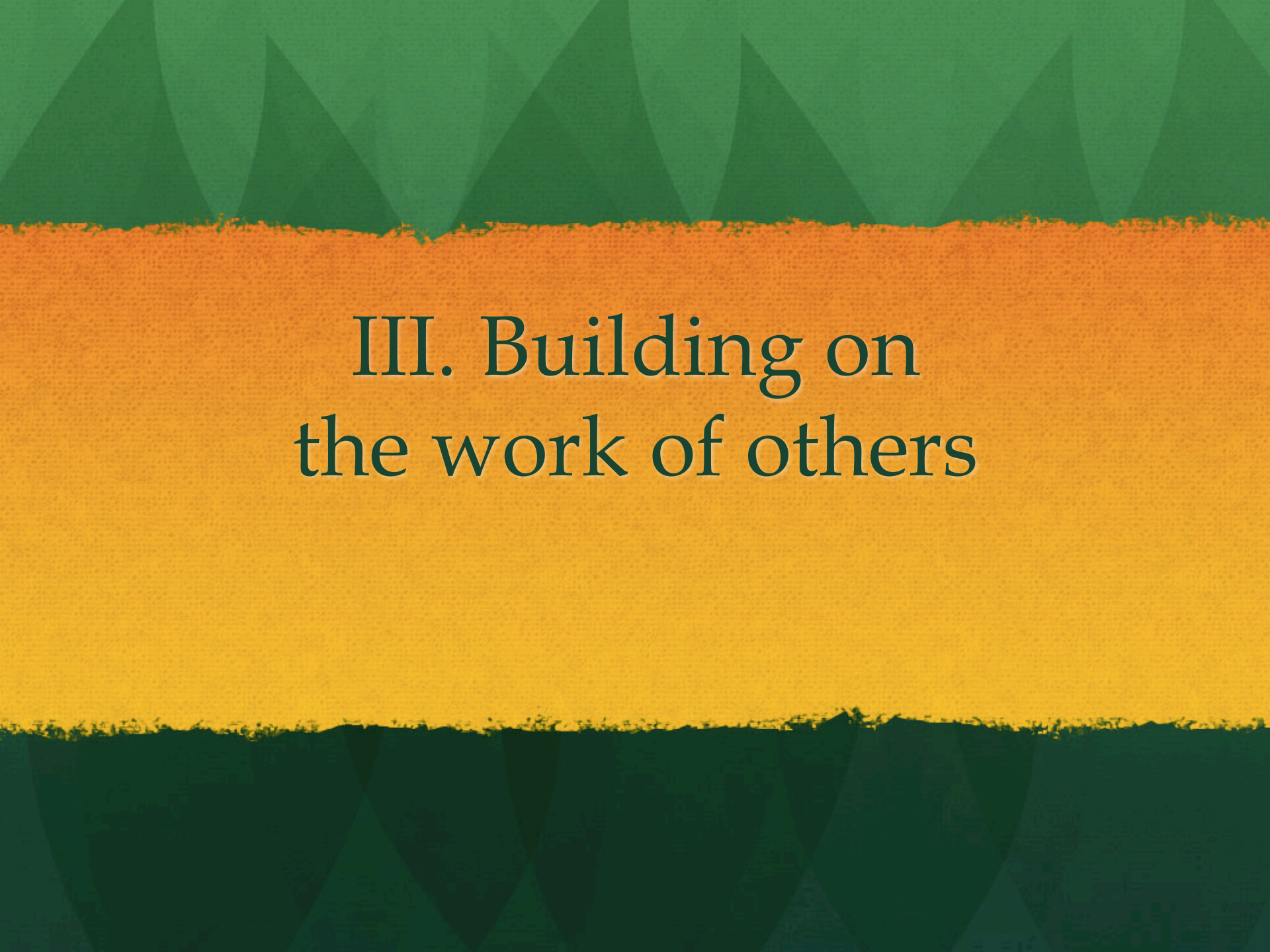
- Training/testing alone won't be enough.
- This will merely be the end of an anomaly.
Designing subtle and informative experiments plays a critical role in physics, in biology, in all experimental sciences.
- The interpretation of experiments depends on how we “think” about the phenomena.

Challenges

- We cannot safely encapsulate learning into traditional software components.
- Experimentation becomes trickier when learning systems become more capable.



- How to “think” about learning systems?
- How to “build on the work of others”?



III. Building on the work of others

The Work of Others

How to package the work of others?

- Digital computers : “software”
- Learning computers :
 - Trained module as a software component?
 - ✘ Trained components do not offer solid “contract”.
 - Training software?
 - ✓ If you can get the training data and replicate the rig.
Recent example : AlexNet.
 - Task specific trained features
 - ✓ Nearly as good.

Example: face recognition

Interesting task: “Recognizing the faces of 10^6 persons.”

- How many labeled images per person can we obtain?

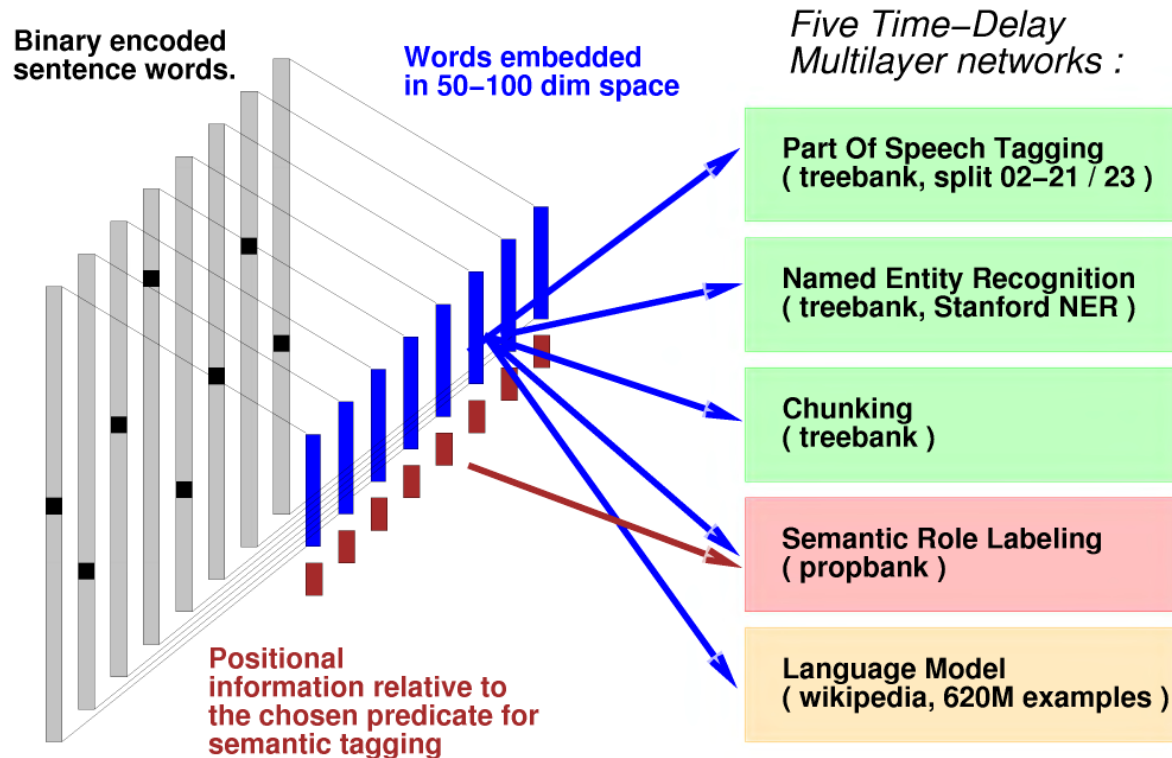
Auxiliary task: “Do these faces belong to the same person?”

- Two faces in the same picture usually are different persons.
- Two faces in successive frames are often the same person.



(Matt Miller, NEC, 2006)

Example: NLP tagging



(Collobert, Weston, et al., 2008-2011)

Example: NLP tagging

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
454	1973	6909	11724	29869	87025
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Example: object recognition

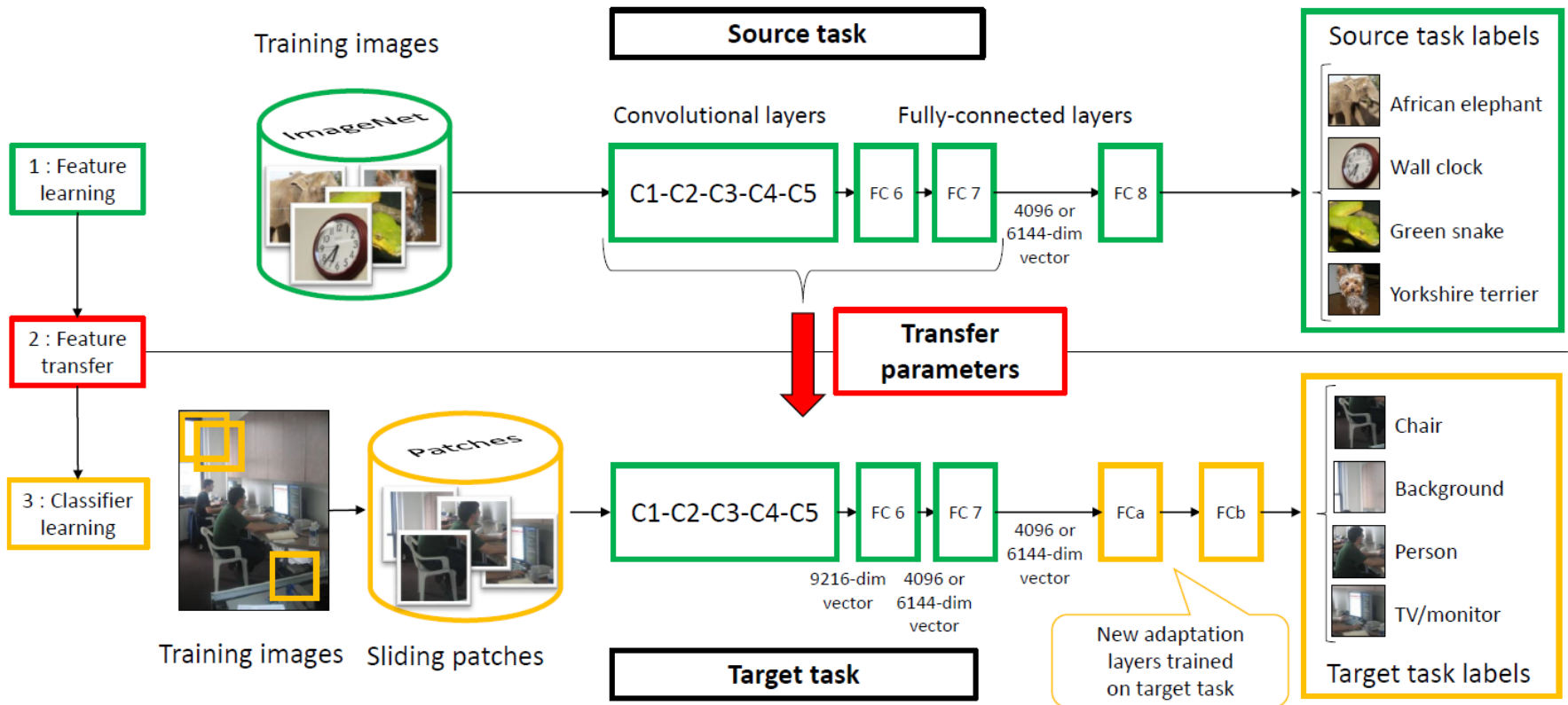
Dogs in ImageNet
($\sim 10^6$ dogs)



Dogs in Pascal VOC
(only $\sim 10^4$ images)



Example: object recognition



(Oquab, B., Sivic, Laptev, CVPR 2014)

Example: object recognition

Held record performance on:

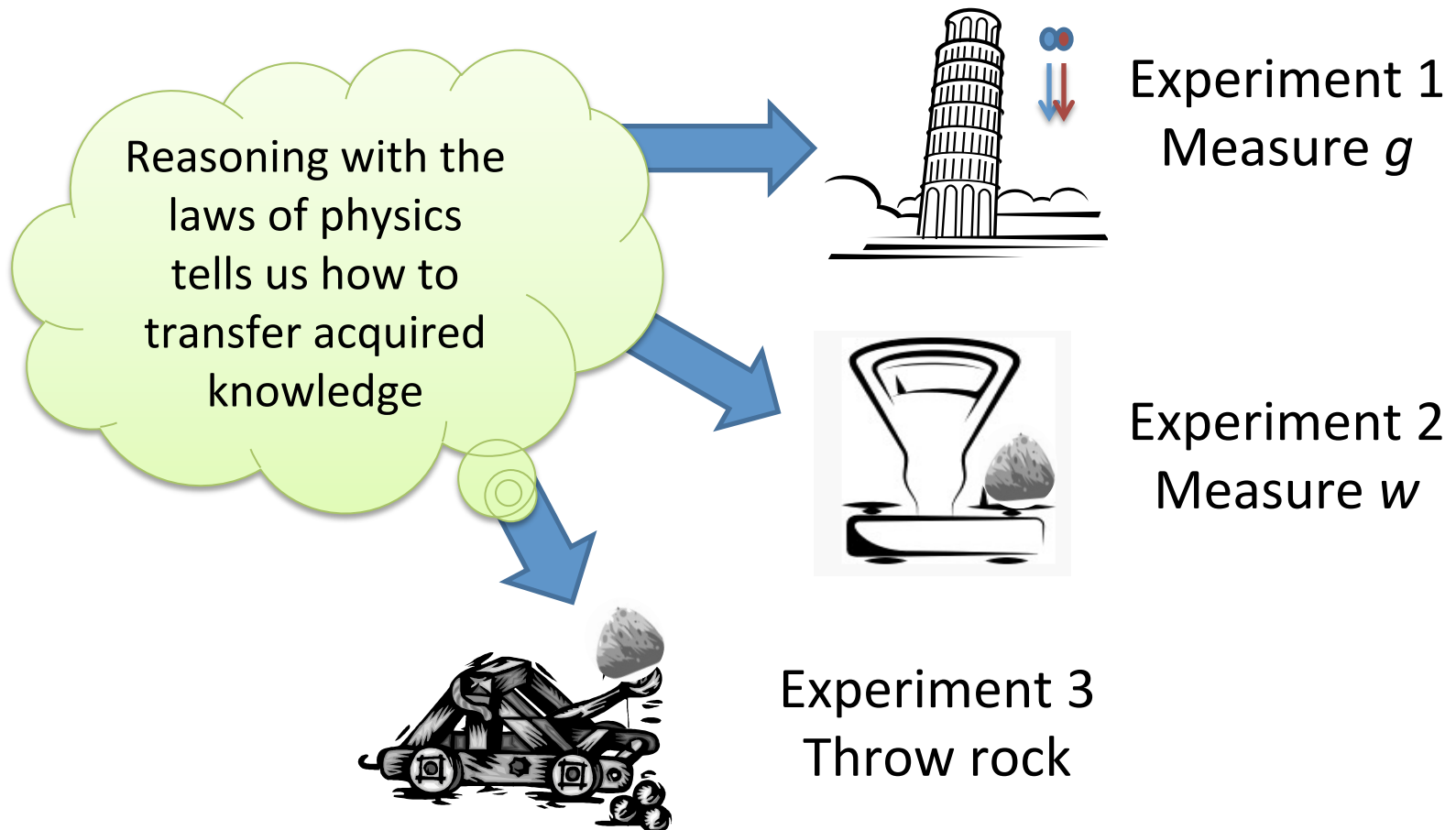
- Pascal VOC 2007 Classification
- Pascal VOC 2012 Classification
- Pascal VOC Action Detection

Comparable works

- Caltech 256 Transfer (Zeiler & Fergus)
- Pascal VOC Detection (Girshick, Donahue, Darrell, Malik)

Feature transfer is becoming the standard in computer vision.

Transfer Learning and Reasoning



Transfer Learning and Reasoning

Reasoning models of various complexities

- Laws of (classical) physics
 - Very sophisticated reasoning model:
First order logic + counterfactuals (Lewis73) + ?
- Feature transfer
 - Minimal reasoning model:
rewiring neural net layers (possibly recursively.)

I prefer to start with the simple one...

Circuit algebra

Rewiring as Algebraic Operation

- Rewiring simultaneous operates in two spaces:
 - ◇ Composition of statistical models.
 - ◇ Composition of model realizations.
- *Transporting the functions and their parametrization*
- Inherited structure in the parameter spaces
- Inherited structure in the “space” of questions of interest

Algebraic structure is an expression of the semantics

- Circuit algebra \iff Semantic Equation Models (Pearl, 2000).
- Causal semantics rather than probabilistic semantics.

Enriching the semantics

Algebraic structure is an expression of the semantics

- Enriching the algebraic structure \iff Enriching the semantics.



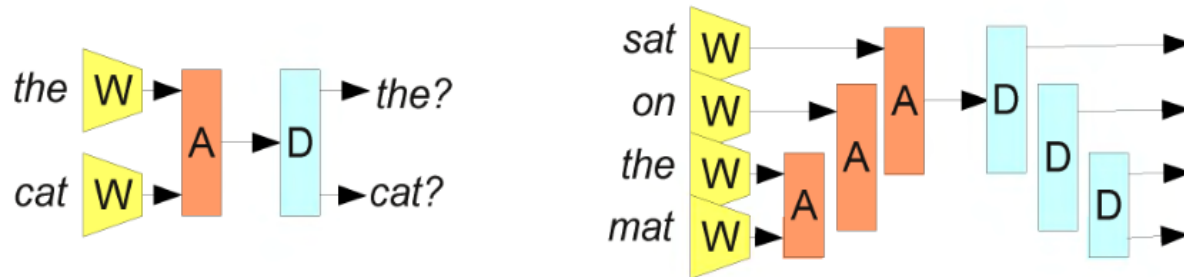
Making the structure recursive

- A time-honored way to generate rich algebraic structures.

Recursive Auto-Associative Memory

Elements

- A representation space \mathcal{R} .
- Association module $A : \mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}$.
- Dissociation module: $D : \mathcal{R} \rightarrow \mathcal{R} \times \mathcal{R}$.



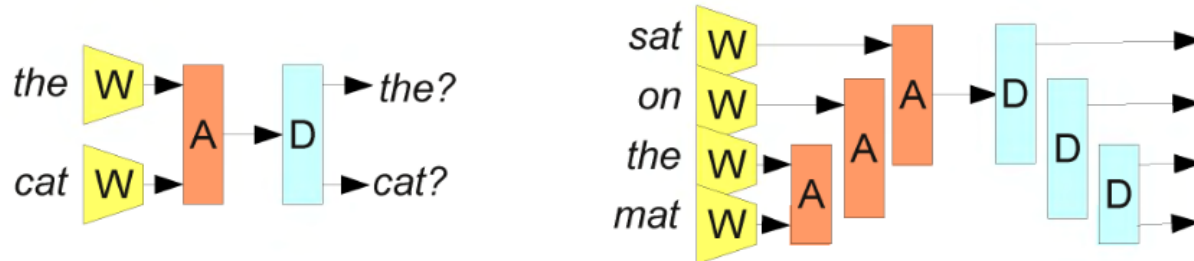
- Desired invariance: $D(A(x, y)) = (x, y)$.

(Pollack, 1988) (Hinton, 1990)

Infinite depth structures

Algebraic structure matters more than representation space \mathcal{R} .

- RAAMs can represent infinite depth predicates.
- Same as cons, car, cdr.



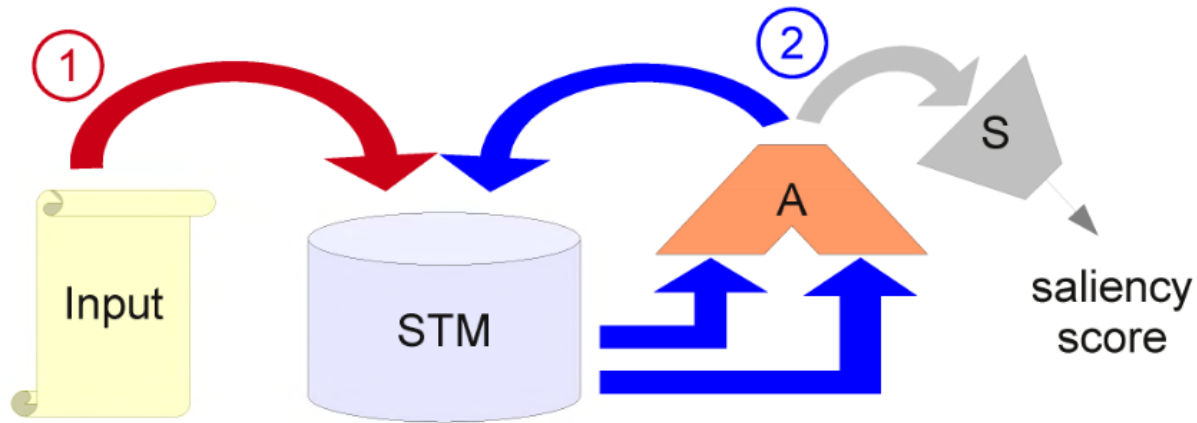
Approximate invariance

- Consider a numerical representation space, i.e. $\mathcal{R} = \mathbb{R}^{100}$.
- Numerical accuracy will eventually degrade reconstruction.
- If the embeddings in the representation space make sense the dissociation module then reconstruct approximate sentences.

Universal Parser?


Elements

- Saliency module $S : \mathcal{R} \rightarrow \mathbb{R}$
- Short term memory.



- Parsing text and images e.g. (Socher, 2010).
- Parsing anything in fact.
- Related to “chunking” (Miller, 1956).

Universal Parser

- ✓ Supervised training of a parser works (Socher, 2010)
-  Weakly supervised training
 - without giving labels that describe the structure,
 - but giving labels for a task that we think needs the structure, does not elicit a meaningful structure!

(Scheible & Schuetze, 2013) : randomly cutting structure did not hurt sentiment analysis

(Etter 2008) : forcing left-to-right groupings did not hurt language models

Left-to-right groupings = recurrent network.

The recurrent network has enough state for implementing the parser.

All hail recurrent networks....

The knowledge of others

Joe knows something (in brain space)
He wants to share it with Jack.



Noisy low bandwidth channel



To deal with noise, one uses
discrete redundant code.
(language space)



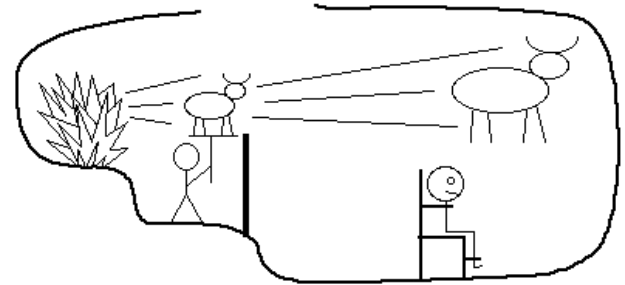
The message elicits adequate representations
in Jack's brain (thanks to common knowledge)

Plato's cave – in the brain

Thoughts happen in brain space.
We cannot observe them directly.
We cannot share them directly.



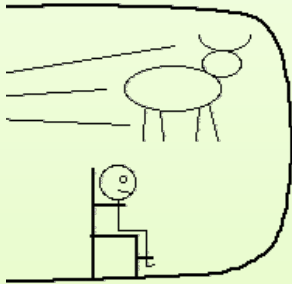
What we observe and share
is the shadow of thought:
language.



Plato's cave – in the brain

Shadows are good enough to support human-like reasoning

- Forget brain space



- Define adequate language
- Define inference principles
- Collect fact and rules.

Shadows are not good enough to support human-like reasoning

Fortunately language is designed to elicit good brain space representations.

- Select task(s) at language level.
- Jointly train
 - Mapping between language space and “thought vectors”
 - Operations on “thought vectors”
- They should implement the task(s) and also respect formal algebraic invariants valid in language space.



Conclusions

Rereading “Perceptrons”

Lessons

- Learning is an experimental science.
 - Experimentation today is simplistic.
 - Learning needs a mathematical backbone that allows us to build on the work of people who preceded us.
 - Algorithmic theory does not cut it.
- + Bonus: Plato’s cave in the brain.

They got this wrong!

but we shouldn't brag

They got this right!

but not that!