Foundations of Artificial Intelligence

CS472/3 — Fall 1999
Lecture #28 & #29

Bart Selman

Today

- wrapup Probably Approximately Correct Learning
  (see also notes #27)
- Neural Network Learning

Technical Aside

In checking derivation of (18.2) of R&N, use the fact that

$(1 - \epsilon) \leq e^{-\epsilon}$ if $0 \leq \epsilon \leq 1$

Examples

1) **H** space of all Boolean functions.
   not PAC learnable; space too big;
   need too many examples.

2) Decision lists.
   PAC learnable.

3) Conjunction of literals.
   PAC learnable.

Examples

4) k-term DNF.

$T_1 \vee T_2 \vee \ldots \vee T_k$, with each T_i conjunction of literals.

not PAC learnable.

Polynomial sample complexity (small $|H|$) but intractable to find consistent hypothesis (most likely).

5) k-CNF is PAC learnable!

Learning a **conjunction of literals** is another example of a PAC learnable language.

E.g. concept: $Old \wedge \neg Tall \wedge Rich \wedge \ldots$

We have: $|H| = 3^n$

Substitution in (1) gives us:

$m \geq \frac{1}{\epsilon}(n\ ln(3) + ln(\frac{1}{\delta}))$

Example: Learn a conjunction with up to 10 literals.

We desire 95% probability that we find an hypothesis with an error of less than 0.1.

How many examples do we need?

$m = \frac{1}{0.1}(10\ ln(3) + ln(\frac{1}{0.05})) = 140$

Surpise: **Not that many!**

Again, the laws of probability...

E.g. in restaurant case we got something with only 12 examples. Figures with up to 100 examples already very good. (Check fig. 18.9. R&N.)

PAC: Concluding Remarks

Valiant showed:

a.) Machines can provable learn certain classes of concepts.

b.) Classes are nontrivial and of interest from a "knowledge representation" perspective.

c.) Process takes only a feasible number of steps (and thus feasible number of examples).
(contrast with previous models)

Some issues to consider:

- Impact on the practice of Machine Learning?
  (consider decision tree learning / backpropagation etc.)
- Connection to reasoning and acting?
- Place for "background" knowledge?

Note: in PAC, background knowledge only through **syntactic** form of concepts.

Major Spinoff from PAC: Boosting

Question: Given a learning algorithm that does only slightly better than **random guessing**, can we make it better?

Surprising answer: **Yes**, we can **boost** its performance to be **almost perfect!**

How? (Schapire 1990) — Basic ideas: run many copies of of the algorithm in a clever way.
Technique is now also popular in practice:
Learn many different decision trees / neural nets / decision lists etc. Let them **vote** for the answer on an unseen case.
Works well on stockmarket data!
Also called: "combining experts"

Slide CS472–11

Concludes PAC learning.

Before going to **Neural Netsworks** let's consider

Question 18.1 R&N

Discuss how infant learns to speak and understand a language.

Possibly most powerful example of learning (if it is learning!).

Slide CS472–12

Issues

Need to: *recognize speech*, *learn vocabulary*,
learn grammar, and learn semantics and pragmatics.

What feedback does child receive?
Enough examples? What kinds of examples?
How many examples sentences?

Slide CS472–13

Reinforcement learning based on general well-being?
Possible natural tendency for “mimicry” essential...
Some direct feedback (parents: “shoe”, “table”, etc.)
But adults do not appear to “correct’ child’s speech!
Also, mainly (only?) positive examples.

Slide CS472–14

Chomsky (1960's) **"poverty of the stimulus"** argument:
basic universal grammar of language **must be innate**.
(simply not enough examples; also no negative ones.)

Recent advances in learning e.g. on **probabilistic context free grammars** re-visits the issue!

Resolution possibly within next 10 years.

Slide CS472–15

Neural Networks

Slide CS472–16

Rich history, starting in the early forties.

(McCulloch and Pitts 1943)

(including at least on suspicious death ...)

Two views:

- **Modeling the brain**.
- **"Just" representation of complex functions.**

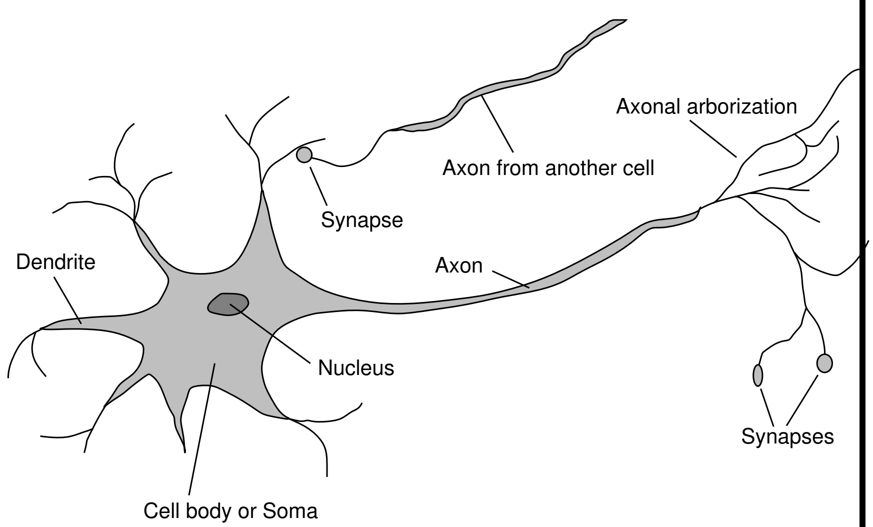
(Continuous; contrast decision trees.)

Much progress on both fronts.

Drawn interests from:

Neuro-science, Cognitive science, AI,
Physics, Statistics, and CS / EE.

Slide CS472–17



Slide CS472–18

Neurons / nerve cells

cell body or **soma**
branches: **dendrited**
single long fiber: **axom**
(100 or more times the diameter of cell body)
axon connects via **synapsis** to dendrites of other cells
signals propagated via complicated electrochemical reaction
each cell has a certain electrical potential
when above **threshold**, pulse is sent down axon

synapses can increase (**excitatory**) / decrease (**inhibitory**) potential (signal)
but most importantly: have **plasticity** — can **learn / remember!**
In fact, learning can happen to single cell!
Note: current model gives neuron with little stucture. Complexity arises out of connectivity. Not clear this is “final” model.

Idea: collection of simple cells leads to complex behavior: *thought, action, and consciousness* ....
Challenged by e.g. Penrose.
Contrast with current computer design.

Massively Parallel

Neurons: highly parallel computation.

10 to 100 steps — given simple timing constraints, one can deduce that certain visual and other cognitive computations are carried out in about 10 to 100 layers of neurons.

Interesting experiments about how visual features we can detect in parallel.

Appears to need **massive parallelism.**

Slide CS472–21

	Computer	Human Brain
Computational units	1 CPU, 10^5 gates	10^{11} neurons
Storage units	10^9 bits RAM, 10^{10} bits disk	10^{11} neurons, 10^{14} synapses
Cycle time	10^{-8} sec	10^{-3} sec
Bandwidth	10^9 bits/sec	10^{14} bits/sec
Neuron updates/sec	10^5	10^{14}

Slide CS472–22

Tempting enterprise:

Design computer modeled after the brain.

Good company: Von Neumann (1958)

The Computer and the Brain

But the **connection machine** was not successful
(Hillis 1989 / Thinking Machines)
64K processors.
What was the problem?

Slide CS472–23

R&N:

The exact way in which the brain enables thought is one of the great mysteries of science.

Much recent progress . . ..

Still, there are skeptics. Especially in CS.

Slide CS472–24

The Skeptic's Position

Related to "levels of abstractions" common in CS.
(less so in EE / Cogn. Sci.)

Consider: Try to figure out how a computer program performing a heap sort works.

Q. How far would you get with a voltmeter? Wiring diagram?
Possibly the wrong level of abstraction!

Could be similar problem in understanding higher cognition using CAT scans!

Still, we let's see what neural net research has achieved.

Artificial Neural Networks

Mathematical abstraction!

We have: **units**, connected by **links**.

Each link has a **weight**.

The primary means for long-term storage. (plasticity)

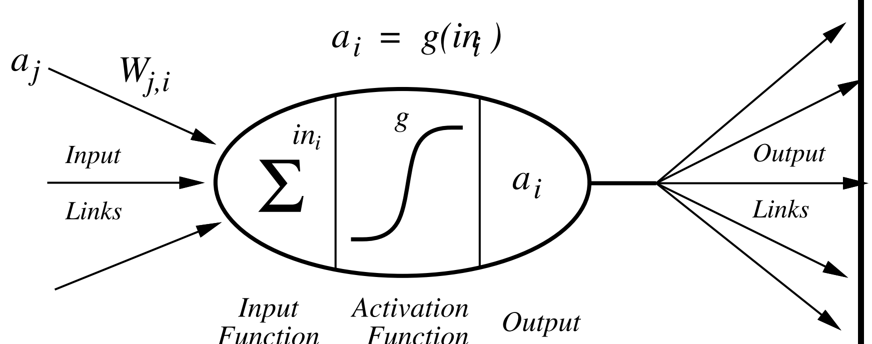
Later: our **learning** algorithms will modify these weights.

What about modifying the connectivity? ("rewiring" the brain . . .).

Each unit has set of inputs links from other units set of output links to other units and an **activation level**.

A means to compute activation level at next time step.

Slide CS472–27



Slide CS472–28

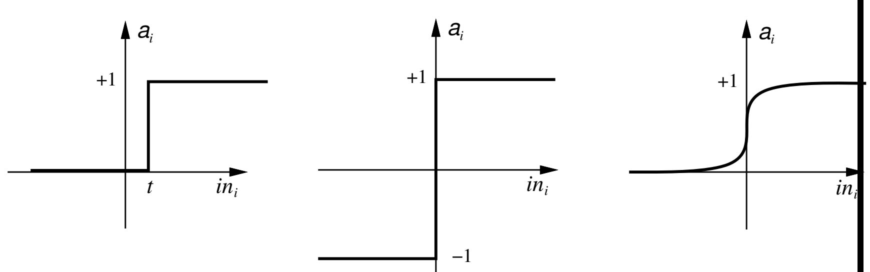
$in_i = \sum_j W_{j,i} a_j$

$a_i \;\leftarrow\; g(in_i) = g(\sum_j W_{j,i} a_j)$

g is the activation function:

step, *sign*, and *sigmoid*.

Slide CS472–29



(a) Step function **(b) Sign function** **(c) Sigmoid function**

Slide CS472–30

a) $step_t(x) = 1$, if $x \geq t$; otherwise 0

b) $sign(x) = +1$, if $x \geq 0$; otherwise -1

c) $sigmoid(x) = \frac{1}{1+e^{-x}}$

What might be the advantage of c?

Slide CS472–31

Aside: Can eliminate thresholds (it's a trick):
create extra input a_0 fixed at -1.

$a_i = step_t(\sum_{j=1}^{n} W_{j,i}a_j) = step_0(\sum_{j=0}^{n} W_{j,i}a_j)$

where $W_{0,i} = t$ and $a_0 = -1$.

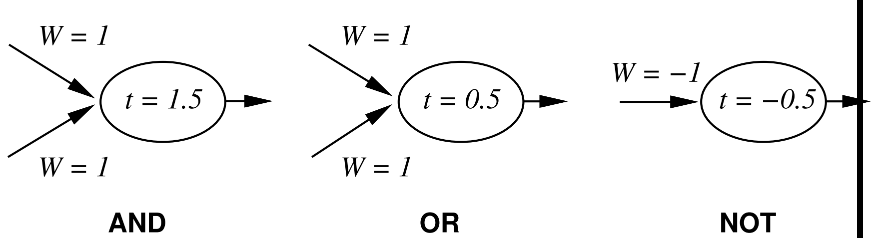
Slide CS472–32

Can simulate Boolean gates!

(original motivation McCulloch and Pitts (1943).

What does this mean?

Slide CS472–33



Slide CS472–34