ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Possiamo definire cos'è un *algoritmo*?

Simone Martini

Dipartimento di Informatica – Scienza e Ingegneria

Un concetto *pre*-matematico

Una sequenza di regole dove
ogni regola
il modo di comporle
sono "effettivi"

La nozione intuitiva è la "pietra di paragone" di ogni definizione formale

La sua formalizzazione perderà (o aggiungerà) sempre
qualche *sfumatura* in rapporto al senso intuitivo

Turing, 1936

Non tratta della nozione di algoritmo,

Definizione di *effettivo*, cioè di *calcolabile in modo meccanico*

effettivo = c'è un algoritmo che lo può calcolare

Ma per una funzione effettiva ci sono molti algoritmi distinti

Molti altri sistemi per funzioni calcolabili:

i sistemi di Post, il lambda-calcolo, o il sistema di Gödel
tutti i linguaggi di programmazione, quali C, Java, Python

Sono tra loro equivalenti: la tesi di Church-Turing

Equivalenza tra i diversi formalismi per la calcolabilità

L'equivalenza è "modulo codifica" (*à codage près*)

Sia S un sistema per la calcolabilità (lambda-calcolo, sistemi di Post, Python, ecc.)

1. C'è in S un modo di "codificare" gli interi: [*n*]
2. Per ogni funzione (Turing-) calcolabile tra interi *f*, c'è un termine F di S che "calcola" *f*

$$F[n] \rightarrow [f(n)]$$

Nessuna garanzia sulla preservazione degli algoritmi!

Lo « or parallelo »

$$por(x,y) = \begin{cases} 0 & \text{si } x \text{ termine ou } y \text{ termine} \\ ne\ termine\ pas & \text{autrement} \end{cases}$$

Calcolabile : valuta "in parallelo" x e y

Lo « or parallelo » nel lambda-calcolo

$$por(x,y) = \begin{cases} 0 & \text{si } x \text{ termine ou } y \text{ termine} \\ ne\ termine\ pas & \text{autrement} \end{cases}$$

Calcolabile : valuta "in parallelo" x e y

G. Berry : **non c'è alcun lambda-termine P** che, per ogni coppia di lambda termini M,N:

P N M ha una forma normale se e solamente se

(almeno) uno tra i due termini M e N ha una forma normale

E tutttavia, il lambda calcolo è Turing-completo !

Lo « or parallelo » nel lambda-calcolo

$$por(x,y) = \begin{cases} 0 & \text{si } x \text{ termine ou } y \text{ termine} \\ ne\ termine\ pas & \text{autrement} \end{cases}$$

Equivalenza "modulo codifica":

- una codifica dei lambda termini (es M e N) con altri lambda termini [M] e [N]
- un'opportuna codifica Eval della procedura di calcolo
- un termine [OR]

tale che

Eval [OR] [M] [N]

ha forma normale se e solo se uno tra M e N ha una forma normale

Algoritmi e livelli di astrazione

Un algoritmo è ben definito solo "modulo un livello di astrazione"
[Gurevich]

La palindromicità di una sequenza: rêver, radar, kayak

Algoritmi e livelli di astrazione

Un algoritmo è ben definito solo "modulo un livello di astrazione"
[Gurevich]

La palindromicità di una sequenza: rêver, radar, kayak

KAYAK

KAYAK

KAYAK

KAYAK

KAYAK

Operazione elementare: "prendere un elemento qualsiasi in una sequenza"

Complessità: lunghezza('KAYAK')/2

Algoritmi e livelli di astrazione

La palindromicità di una sequenza: rêver, radar, kayak

Operazione elementare: "prendere un elemento qualsiasi in una sequenza"
Complessità: lunghezza('KAYAK')/2

La macchina di Turing non può
"prendere un elemento qualsiasi in una sequenza" !

Avanti e indietro ogni volta!

RESSASSER

Complessità: lunghezza('RESSASSER')^2

Algoritmi e programmi

Al limite, si rischia di identificare algoritmi e programmi

Per l'informatica sono cose diverse:

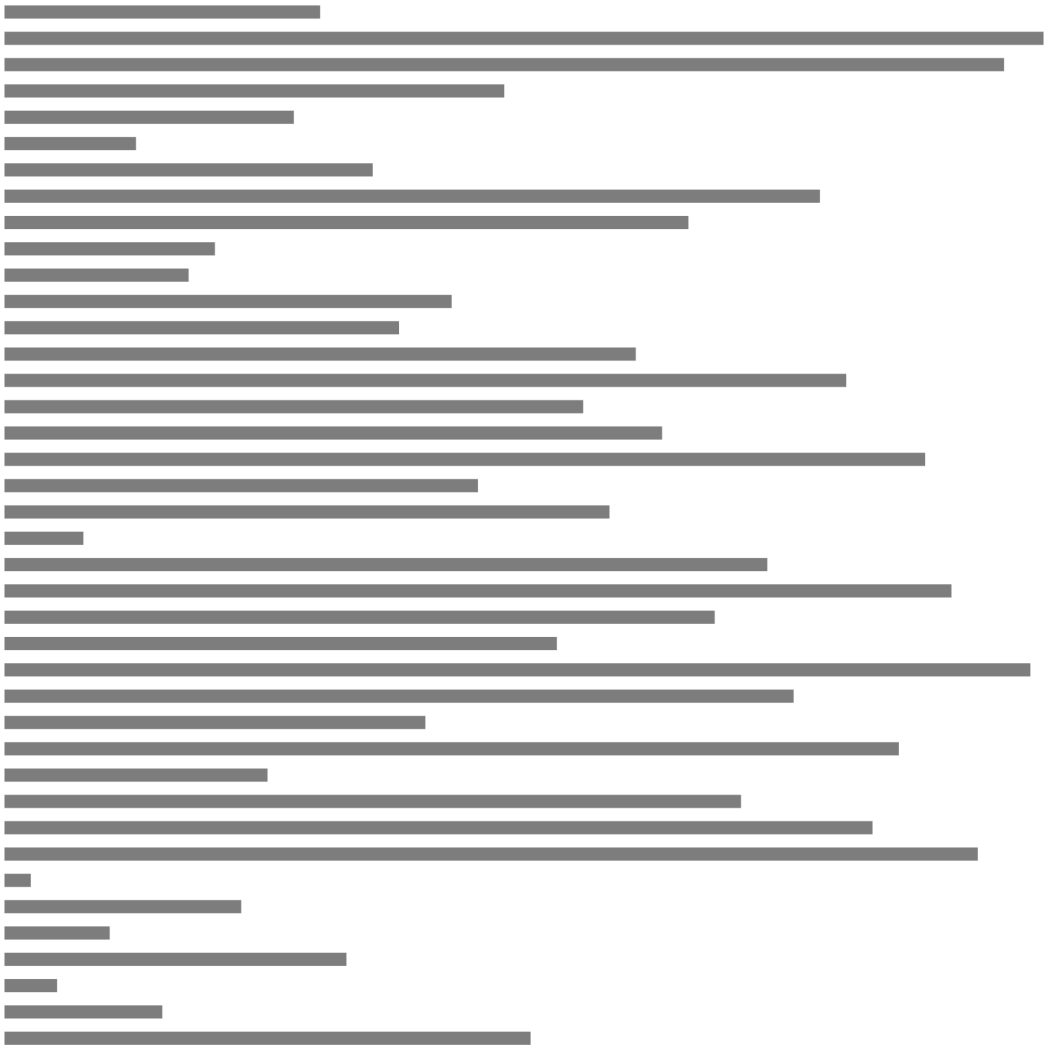
- la descrizione generica di un procedimento;
- la sua "traduzione", la sua "codifica", in un linguaggio di programmazione

E tuttavia,

due linguaggi non hanno mai lo stesso insieme di operazioni elementari sui dati…

Algoritmi e programmi: *Quicksort*

Quicksort [Hoare, 1961]



Algoritmi e programmi: *Quicksort*

Python:

```
def QuickSort(L):
  if L==[]: return L
  pivot = L[0]
  return QuickSort([x for x in L[1:] if x < pivot])
         + [pivot] +
         QuickSort([x for x in L[1:] if x >= pivot])
```

Algoritmi e programmi: *Quicksort*

JAVA:

```
public static void quickSort(int[] arr, int start, int end){
        int partition = partition(arr, start, end);
        if(partition-1>start) {
            quickSort(arr, start, partition - 1);
        }
        if(partition+1<end) {
            quickSort(arr, partition + 1, end);
        }
}
public static int partition(int[] arr, int start, int end){
        int pivot = arr[end];

        for(int i=start; i<end; i++){
            if(arr[i]<pivot){
                int temp= arr[start];
                arr[start]=arr[i];
                arr[i]=temp;
                start++;
            }
        }
        int temp = arr[start];
        arr[start] = pivot;
        arr[end] = temp;
        return start;
}
```

Python:

```
def QuickSort(L):
  if L==[]: return L
  pivot = L[0]
  return QuickSort([x for x in L[1:] if x < pivot])
         + [pivot] +
         QuickSort([x for x in L[1:] if x >= pivot])
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Algoritmi e programmi: *Quicksort*

JAVA:

```
public static void quickSort(int[] arr, int start, int end){
        int partition = partition(arr, start, end);
        if(partition-1>start) {
            quickSort(arr, start, partition - 1);
        }
        if(partition+1<end) {
            quickSort(arr, partition + 1, end);
        }
}
public static int partition(int[] arr, int start, int end){
        int pivot = arr[end];

        for(int i=start; i<end; i++){
            if(arr[i]<pivot){
                int temp= arr[start];
                arr[start]=arr[i];
                arr[i]=temp;
                start++;
            }
        }
        int temp = arr[start];
        arr[start] = pivot;
        arr[end] = temp;
        return start;
}
```

Sono davvero due codifiche diverse dello *stesso* algoritmo ?

Python:

```
def QuickSort(L):
  if L==[]: return L
  pivot = L[0]
  return QuickSort([x for x in L[1:] if x < pivot])
         + [pivot] +
         QuickSort([x for x in L[1:] if x >= pivot])
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

programmi/algoritmi di apprendimento neuronale

Due attori:

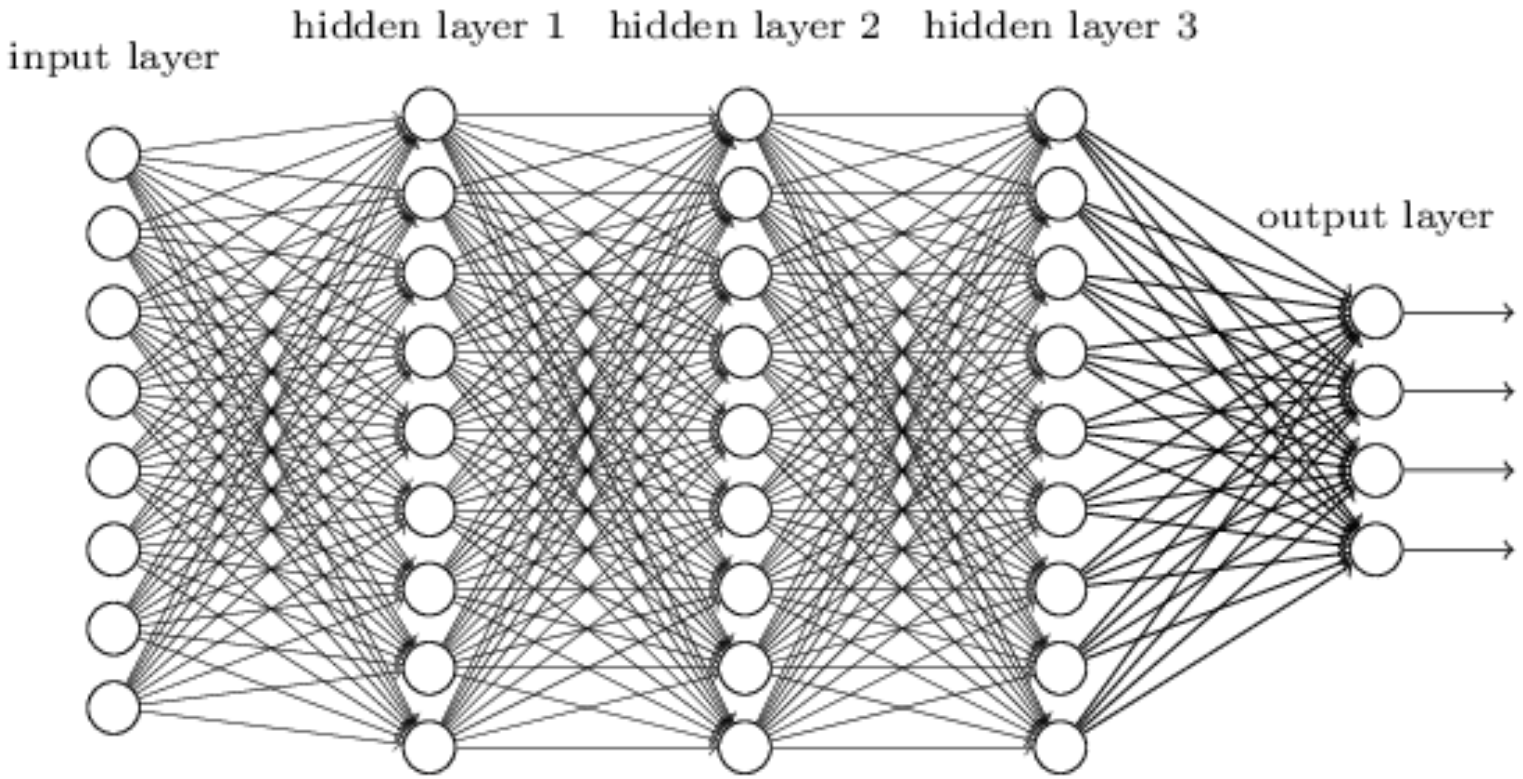
- l'algoritmo di apprendimento (generico)
- il risultato dell'apprendimento
  (il comportamento della rete, dopo l'apprendimento)

L'apprendimento specializza (attraverso l'allenamento)
una rete generica in una funzione particolare

programmi/algoritmi di apprendimento neuronale

Due attori::

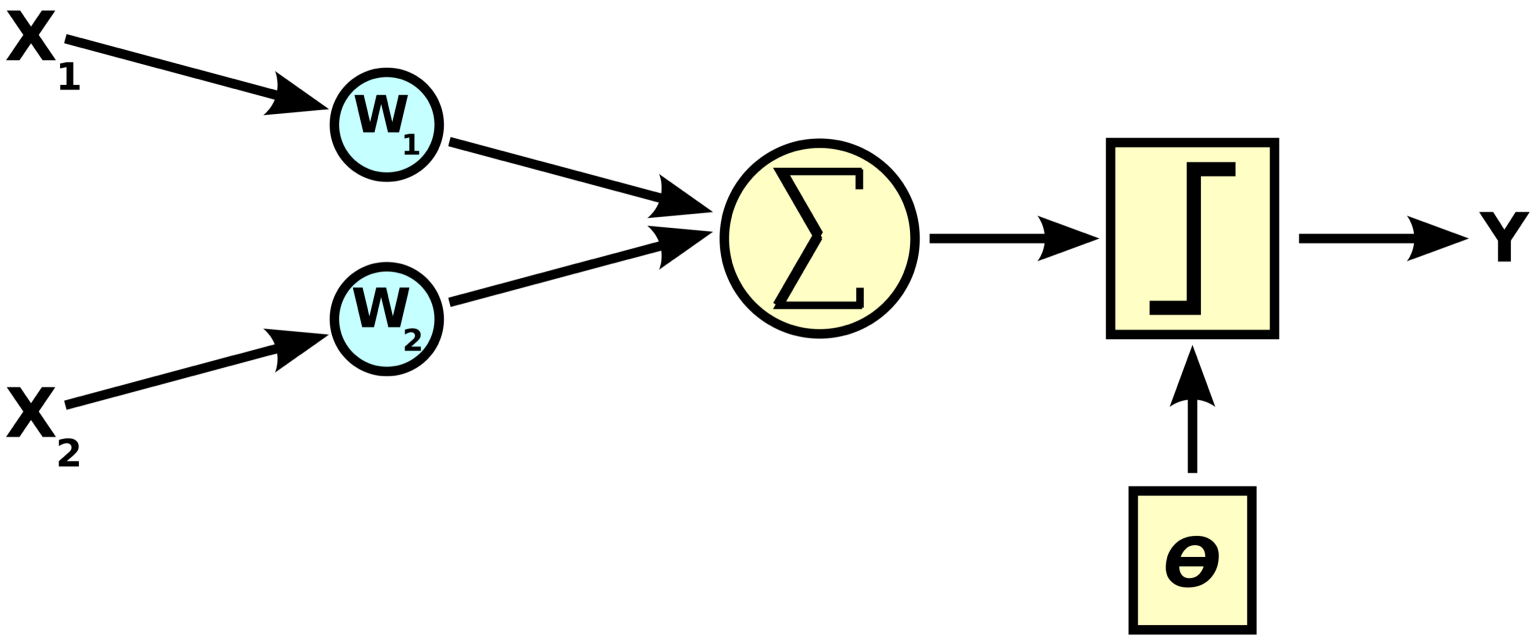
- l'algoritmo di apprendimento (generico)
- il risultato dell'apprendimento
  (il comportamento della rete, dopo l'apprendimento)



programmi/algoritmi di apprendimento neuronale

Due attori::

- l'algoritmo di apprendimento (generico)
- il risultato dell'apprendimento
  (il comportamento della rete, dopo l'apprendimento)



programmi/algoritmi di apprendimento neuronale

L'apprendimento specializza (attraverso l'allenamento)
una rete generica in una funzione particolare

Tutto il risultato dell'apprendimento è contenuto nei *pesi* della rete

Algoritmo « classico » : le scelte fatte dall'algoritmo sono esplicite

« Algoritmo neuronale » : tutto è "opaco", nascosto nei pesi

Una questione di *accountability…*

algoritmi che non terminano

Algoritmi classici: terminazione, « il risultato »

Un sistema operativo : un ciclo infinito che *fa cose* attraverso l'interazione con gli attori della computazione (risorse, processi, ambienti, persone, ecc.)

Non trasformazione di dati di ingresso in un risultato,
ma interazione, funzione dei dati, del tempo, degli attori umani…

Il messaggio di questa chiaccherata

Un concetto *intuitivo:* è normativo, pietra di paragone

sequenza di passi effettivi

Non c'è *una sola* formalizzazione

del concetto di algoritmo

di uno stesso algoritmo

Ci sono *varie* formalizzazioni

a vari livelli di astrazione

ciascuna di essa rende conto di alcuni aspetti, e ne perde altri

Il messaggio di questa chiaccherata

la pluralità degli approcci formali

perfino, *cum grano salis*, il loro disaccordo

è un bene

come il pluralismo delle idee in una società democratica sana

Gli algoritmi nella nostra vita quotidiana

- *conoscenza* : l'algoritmo deve essere pubblico!

- *responsabilità* : chi è il responsabile delle decisioni?

- *comprensione* : quali competenze sono necessarie ?

E.g., Maël Pégny, Issam Ibnouhsein.

Quelle transparence pour les algorithmes d'apprentissage machine?
Rev. d'Intelligence Artif. 32(4): 447-478 (2018).

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Simone Martini

Dipartimento di
Informatica – Scienza e Ingegneria

simone.martini@unibo.it

www.unibo.it