Artificial Neural Networks: a new way to solve big data problems

Michele Piccinno

June 24, 2016

Why ANN?

Because of (Alpha)Go!

AlphaGo

Google DeepMind

Standard approach

Algorithm $\Rightarrow$ Steps

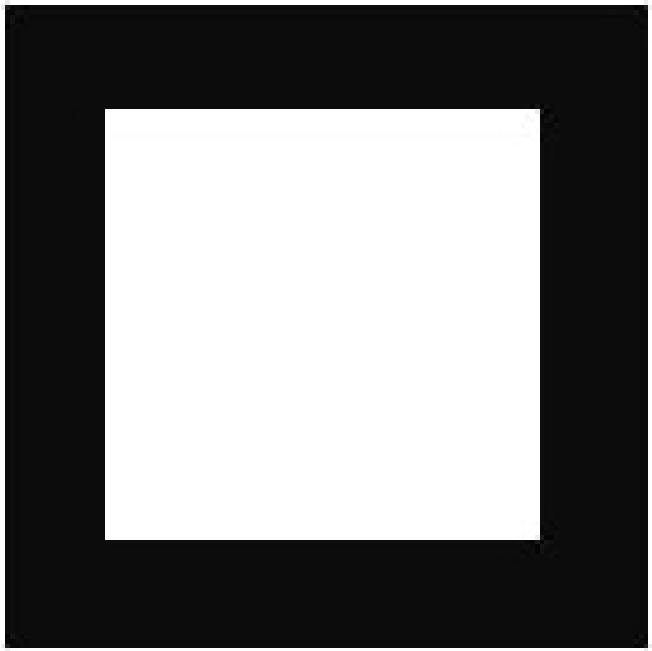
Standard approach

Algorithm $\Rightarrow$ Steps

Heuristic $\Rightarrow$ Guidelines

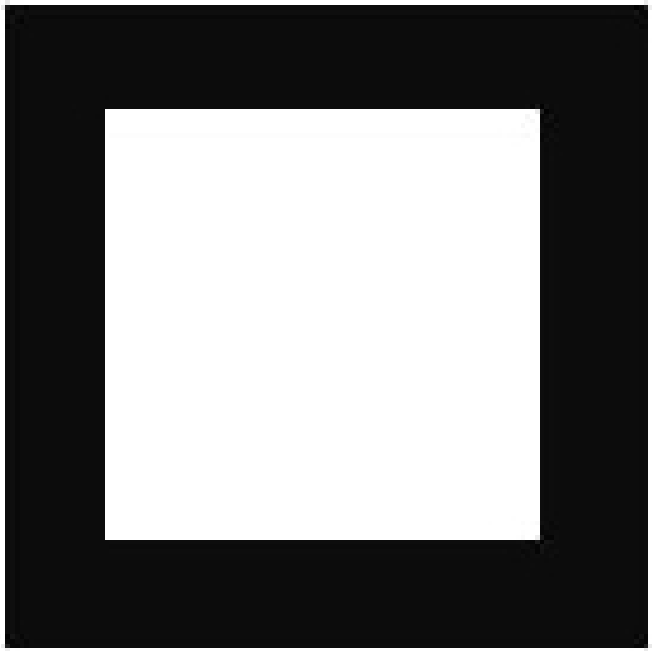
Human approach

What is this?

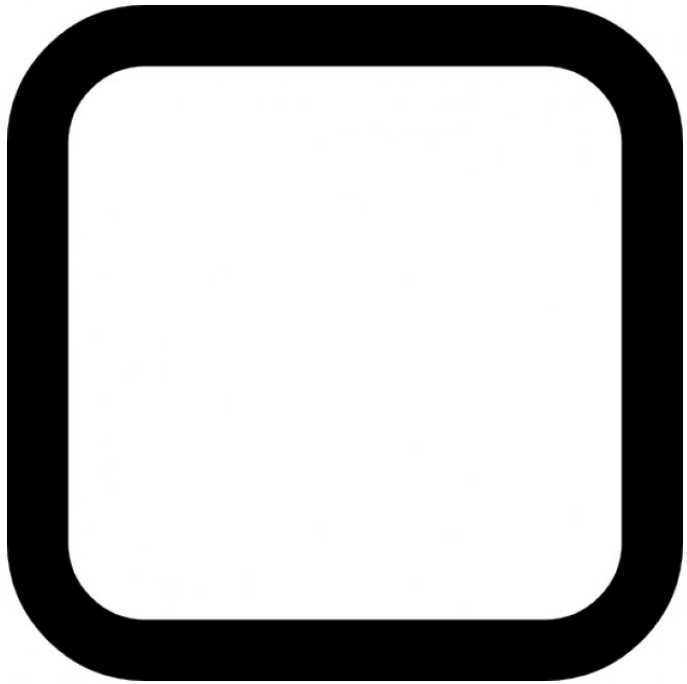


Human approach

What is this?



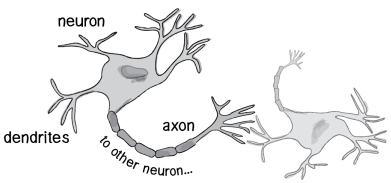
And this?



Neurons

Biological

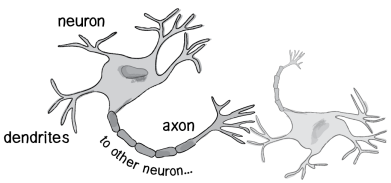
- grouped



Neurons

Biological

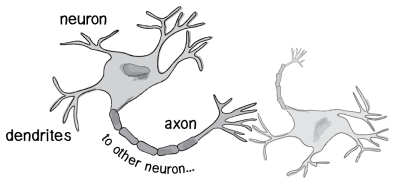
- grouped
- the soma carry out a weighted sum



Neurons

Biological

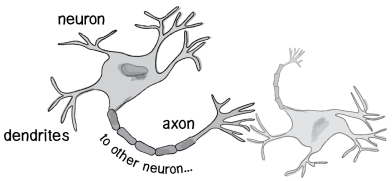
- grouped
- the soma carry out a weighted sum
- if the value is higher than a treshold, neuron shoots



Neurons

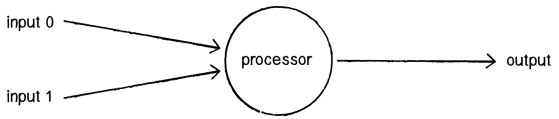
Biological

- grouped
- the soma carry out a weighted sum
- if the value is higher than a treshold, neuron shoots



Artificial

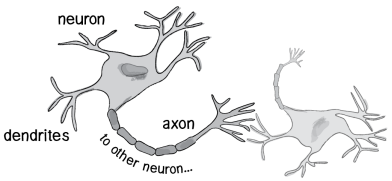
- non-linear net of statistical data



Neurons

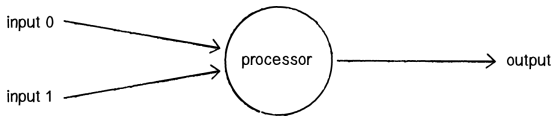
Biological

- grouped
- the soma carry out a weighted sum
- if the value is higher than a treshold, neuron shoots



Artificial

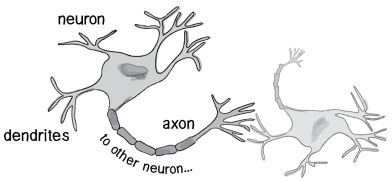
- non-linear net of statistical data
- interconnected



Neurons

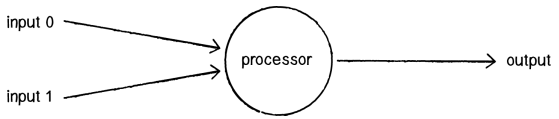
Biological

- grouped
- the soma carry out a weighted sum
- if the value is higher than a treshold, neuron shoots



Artificial

- non-linear net of statistical data
- interconnected
- software and hardware



History

- 1943: first model by McCulloch and Pitts
- 1949: unsupervised learning by Hebb (Hebbian learning)
- 1958: perceptron by Rosenblatt
- 1969: XOR and processing power issues by Minsky and Papert
- 1975: back-propagation by Werbos

Perceptron

Definition

It is a type of linear classifier, that can decide whether an input (represented by a vector of numbers) belongs to one class or another.

$$f(x) = \chi(\langle \omega, x \rangle + b)$$

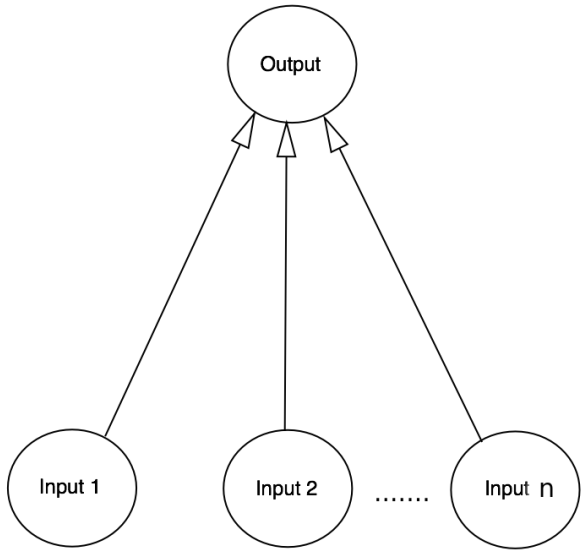
x input vector
$f(x)$ output
χ output function
ω vector of weights
b bias

A simple ANN 1/2

One perceptron iteratively "trained"

$$\omega^{t+1} = \omega^t + \alpha(g(x^t) - f(x^t))x^t$$

x^t t-th input vector
$f(x^t)$ estimated output
$g(x^t)$ real output
ω vector of weights
α learning constant



A simple ANN 2/2

```
from numpy import exp, array, random, dot
training_set_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
training_set_outputs = array([[0, 1, 1, 0]]).T
random.seed(1)
synaptic_weights = 2 * random.random((3, 1)) - 1
for iteration in xrange(10000):
    output = 1 / (1 + exp(-(dot(training_set_inputs, synaptic_weights))))
    synaptic_weights += dot(training_set_inputs.T, (training_set_outputs - output) * output * (1 - output))
print 1 / (1 + exp(-(dot(array([1, 0, 0]), synaptic_weights))))
```

	Input			Output
Example 1	0	0	1	0
Example 2	1	1	1	1
Example 3	1	0	1	1
Example 4	0	1	1	0
New	1	0	0	?

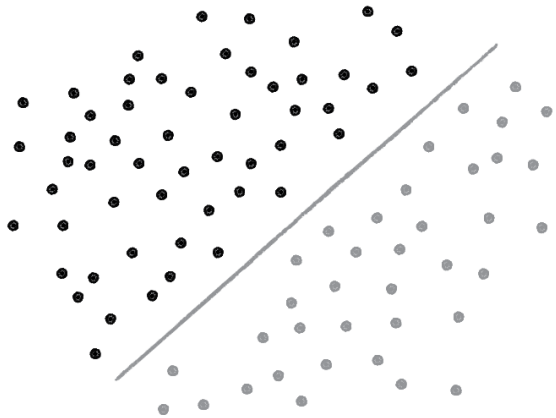
A simple ANN 2/2

```
from numpy import exp, array, random, dot
training_set_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
training_set_outputs = array([[0, 1, 1, 0]]).T
random.seed(1)
synaptic_weights = 2 * random.random((3, 1)) - 1
for iteration in xrange(10000):
    output = 1 / (1 + exp(-(dot(training_set_inputs, synaptic_weights))))
    synaptic_weights += dot(training_set_inputs.T, (training_set_outputs - output) * output * (1 - output))
print 1 / (1 + exp(-(dot(array([1, 0, 0]), synaptic_weights))))
```

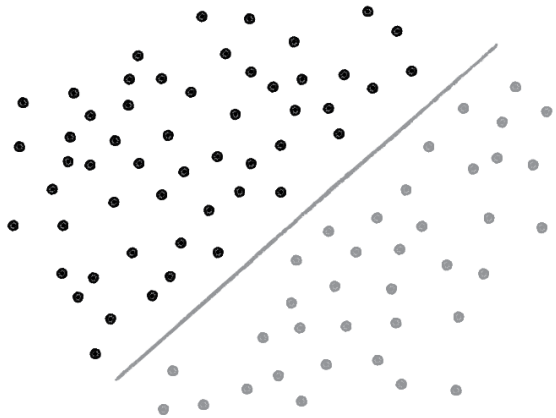
	Input			Output
Example 1	0	0	1	0
Example 2	1	1	1	1
Example 3	1	0	1	1
Example 4	0	1	1	0
New	1	0	0	?

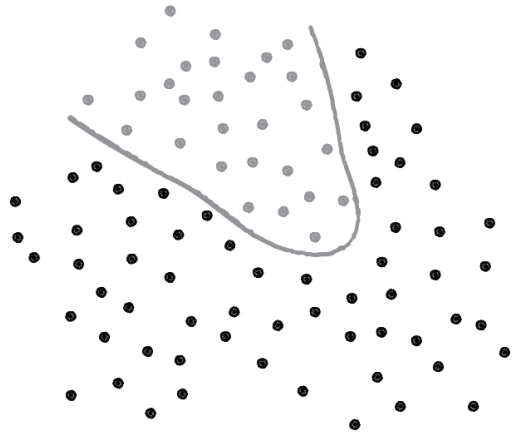
```
Random starting synaptic weights:
[[-0.16595599]
 [ 0.44064899]
 [-0.99977125]]

New synaptic weights after training:
[[ 9.67299303]
 [-0.2078435 ]
 [-4.62963669]]

Considering new situation [1, 0, 0] -> ?:
[ 0.99993704]
```

Linear separability

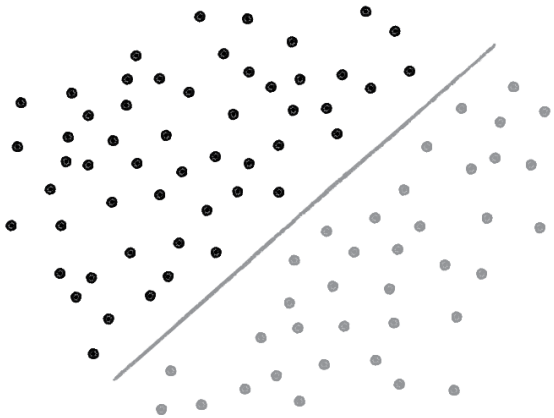


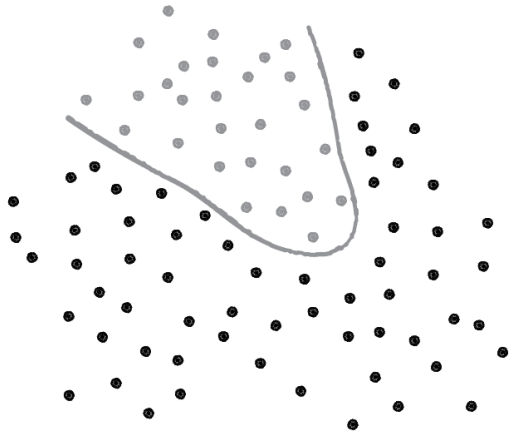
Linear separability

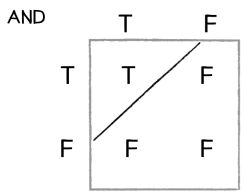


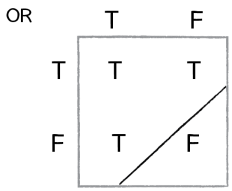


Linear separability

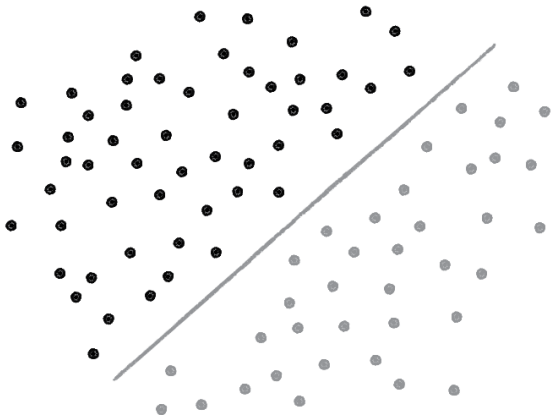


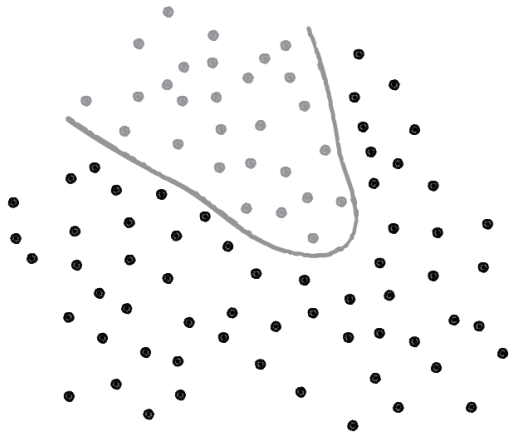


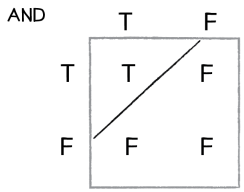


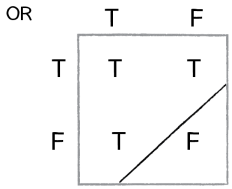


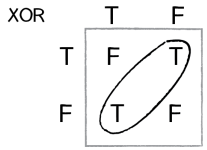
Linear separability





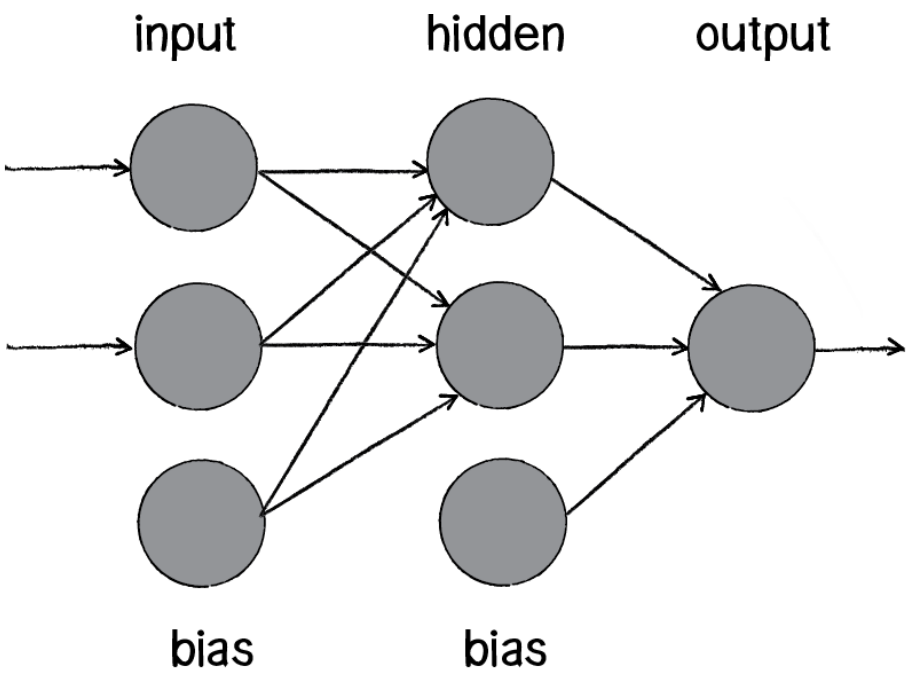






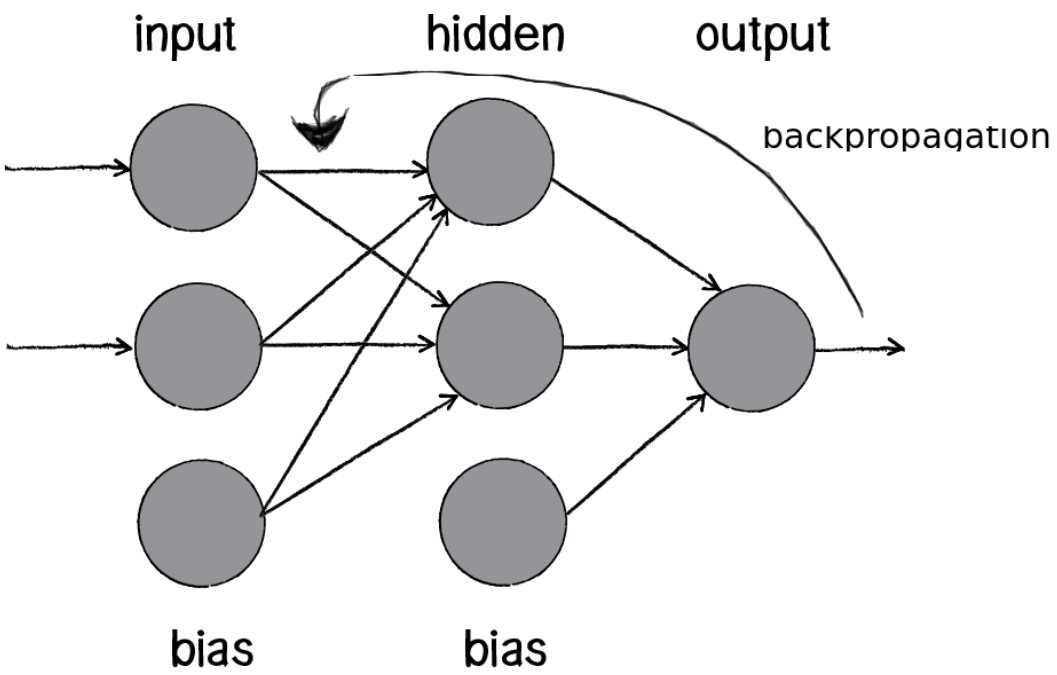
XOR solution

$$\text{XOR} = \text{OR} + \text{NOT AND}$$



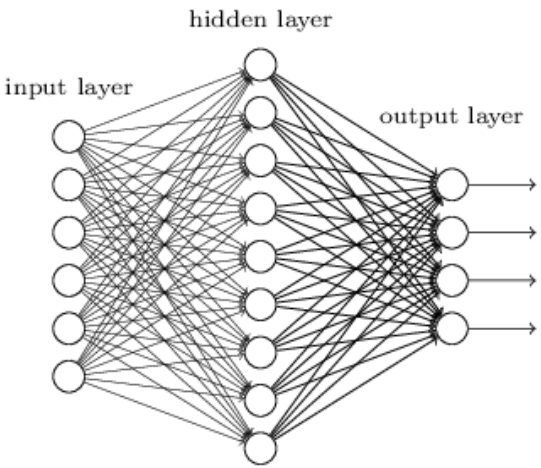
XOR solution

$$\text{XOR} = \text{OR} + \text{NOT AND}$$

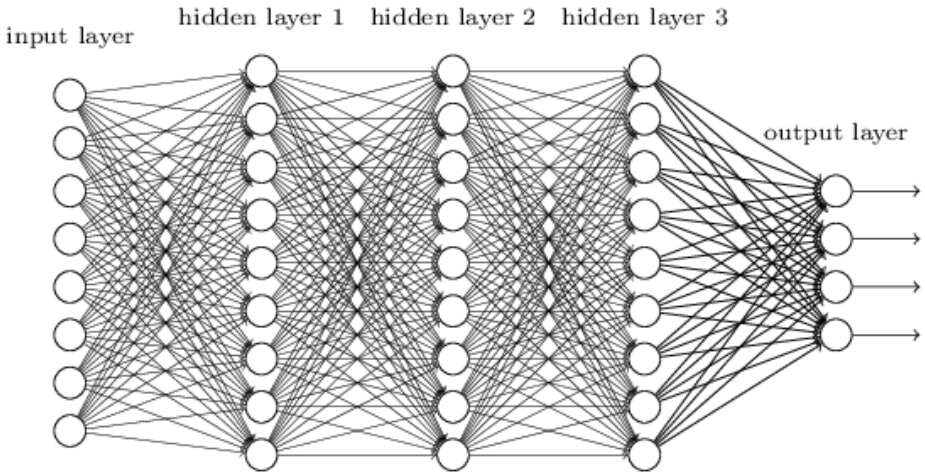


Deep Learning

"Non-deep" feedforward neural network



Deep neural network



High Energy Physics

- BaBar
- CDS
- D0

- track reconstruction
- particle identification and discrimination
- jet tagging

Medical Physics

- Diagnostic field
- Nuclear Imaging

- lung carcinoma
- track reconstruction
- metastasis diagnosis

Conclusion

PROs:

- solve a lot of problems
- versatility
- robust against noise

CONs:

- long to train
- versatility
- black box

Main reference

Daniel Shiffman, *The Nature of Code*
http://natureofcode.com/book/chapter-10-neural-networks/

A neural network is the second best way to solve any problem.
The best way is to actually understand the problem.

Thank you for your attention