Empirical Decision Model Learninng

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Michele Lombardi
michele.lombardi2@unibo.it
Michela Milano
michela.milano@unibo.it

Alessio Bonfietti
Stefano Gualandi
Tias Guns
Andrea Micheli
Andrea Bartolini
Luca Benini
Pascal Van Hentenryck

What makes a problem complex?

A Case Study: Traffic Light Placement

- Add/remove traffic lights in a city
- Traffic lights can be connected (green wave)
- Every operation has a cost
- Budget limit
- **Objective:** improve traffic flow

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

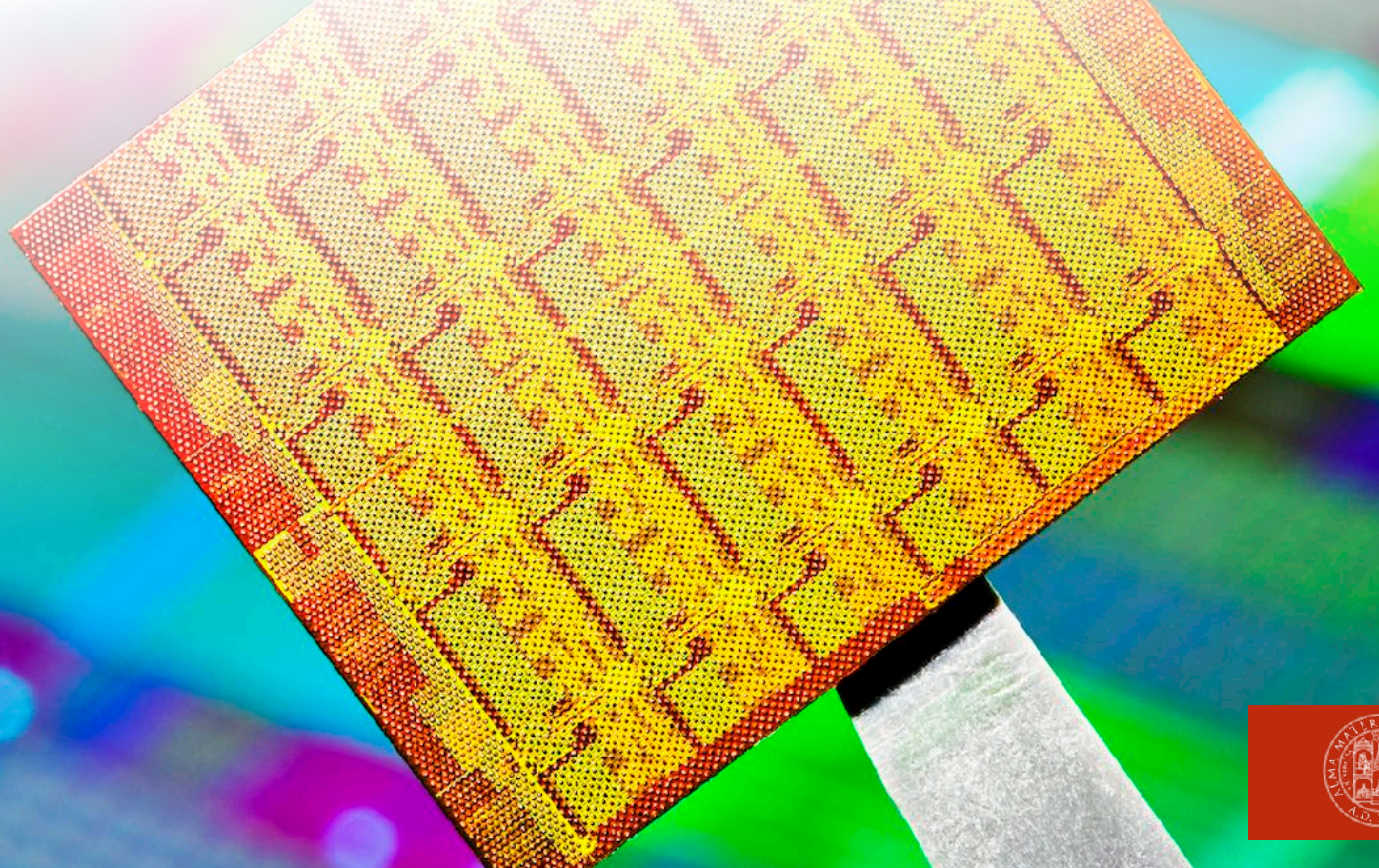
A Case Study: Energy Incentive Design

- Assign resources to incentive actions
- Reach a renewable generation quota
- **Objective**: minimize cost

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

A Case Study: Thermal Aware Job Allocation

- Many-core CPU (Intel SCC, 2009, 48 cores)
- Dispatch jobs
- Load balancing constraints
- **Objective:** avoid thermal hot-spots (efficiency loss)



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

What Makes a Problem Complex?

In general, many things:

- Scale
- Different types of decisions
- Poor bounds/propagation…

But for these problems it's something else!

How do we model…

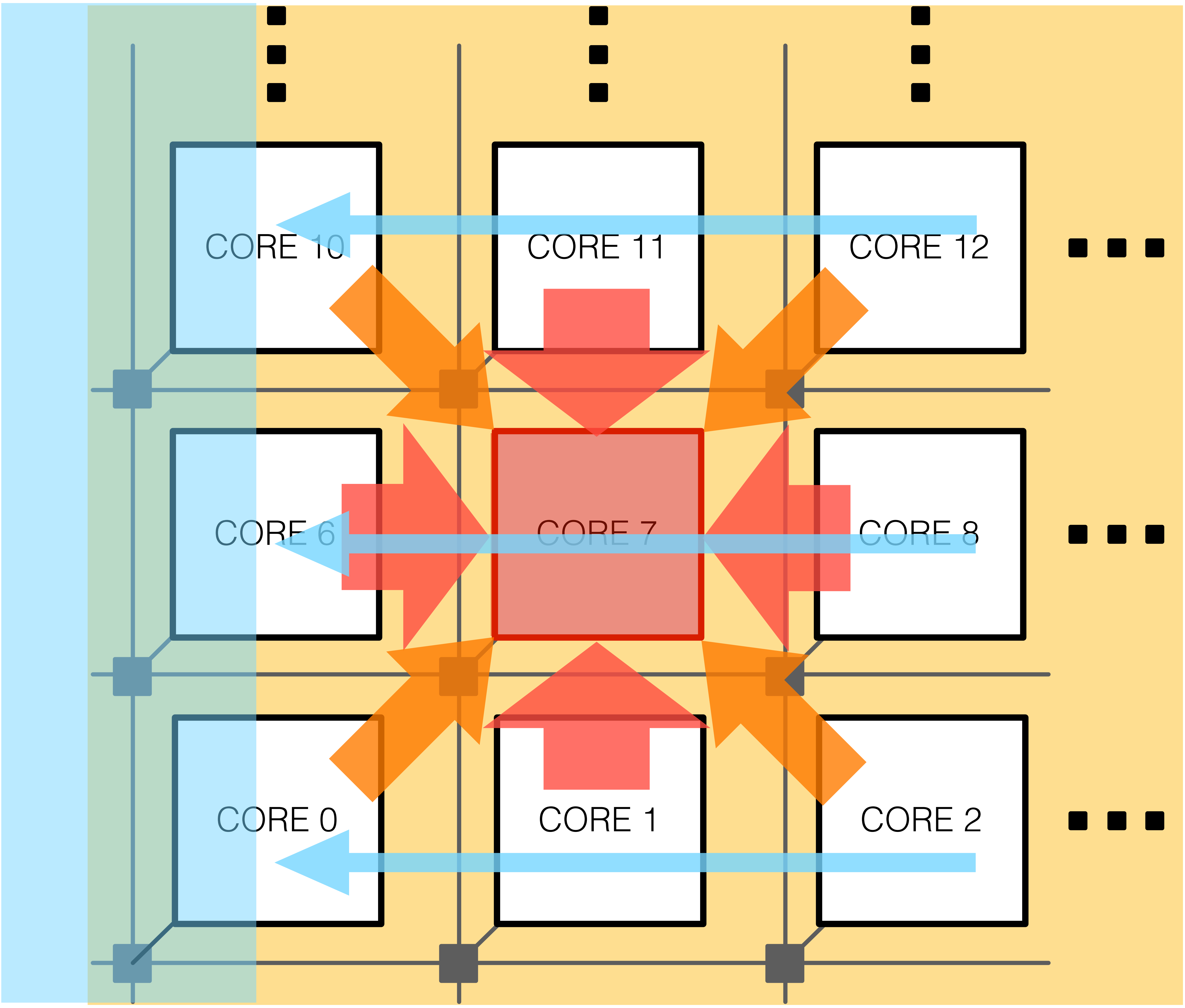
- The link between traffic light location and traffic?
- Between incentives and renewables diffusion/acceptance?
- Between job placement and temperature/efficiency?

This is very hard to do via an expert-driven approach!

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Example: Thermal Aware Job Mapping

The temperature/efficiency of a core is affected by:



- the room temperature
- the workload of each core
- the neighbor workload
- the heat sink positions…

A simulator is viable, but
not so a declarative model

Sometimes, you don't even have a simulator!

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

A possible solution:

Empirical (Decison) Model Learning

Empirical (Decision) Model Learning

- Start from observations

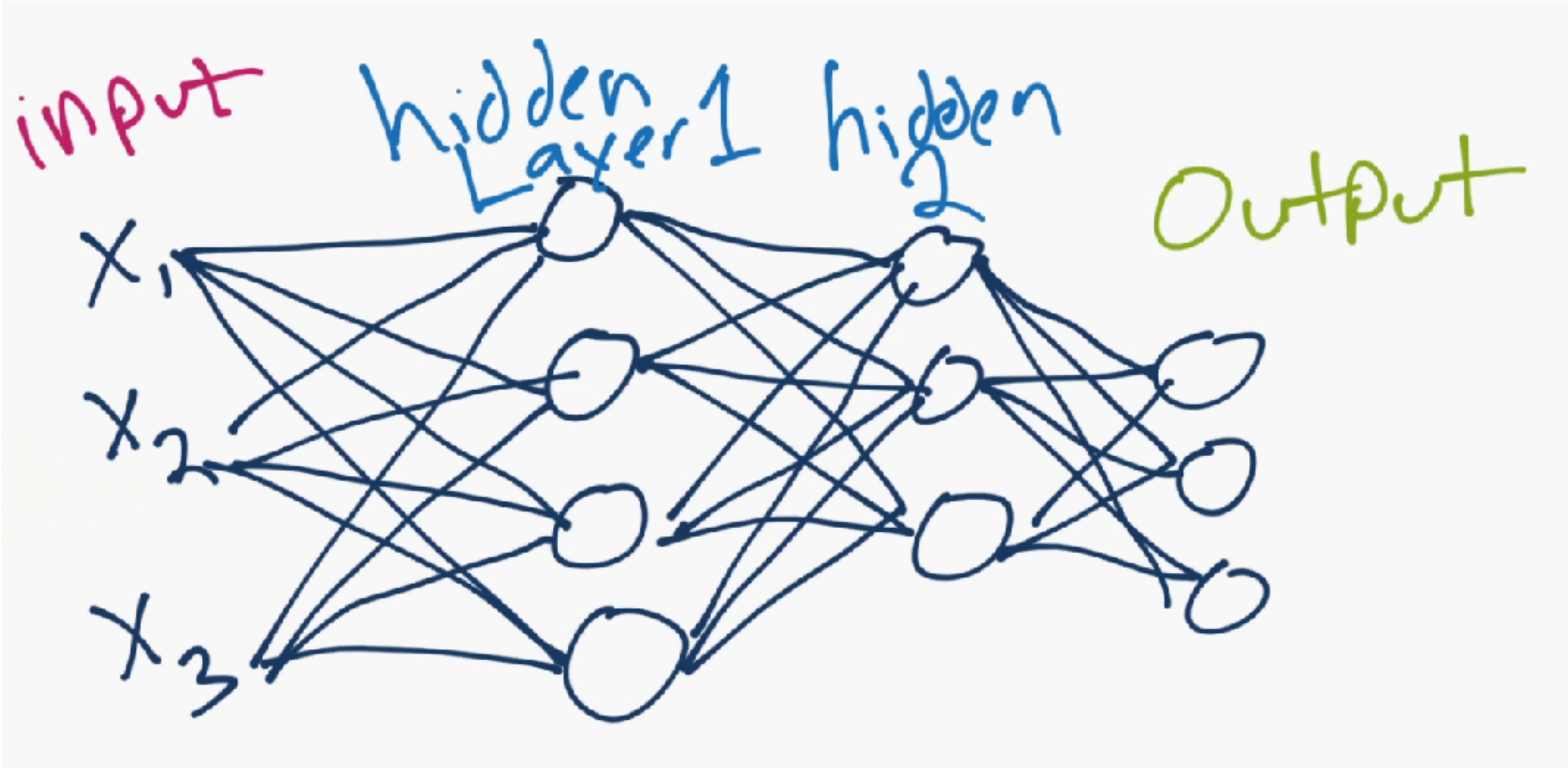
Avg. Load 0	Std. Load 0	Avg. Load 1	Std. Load 1	…
0.9	0.1	0.7	0.3	…
0.8	0.2	0.8	0.1	…
0.5	0.4	0.6	0.2	…
…	…	…	…	…

Core 0	Core 1	Core 2	…
0.9	0.7	0.8	…
0.7	0.9	0.9	…
0.8	0.6	0.8	…
…	…	…	…

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Empirical (Decision) Model Learning

- Start from observations
- Approximate via Machine Learning



$$h : \text{load stats} \mapsto \text{core } k \text{ eff.}$$

Empirical (Decision) Model Learning

- Start from observations
- Approximate via Machine Learning
- Embed the "empirical model" in the combinatorial model

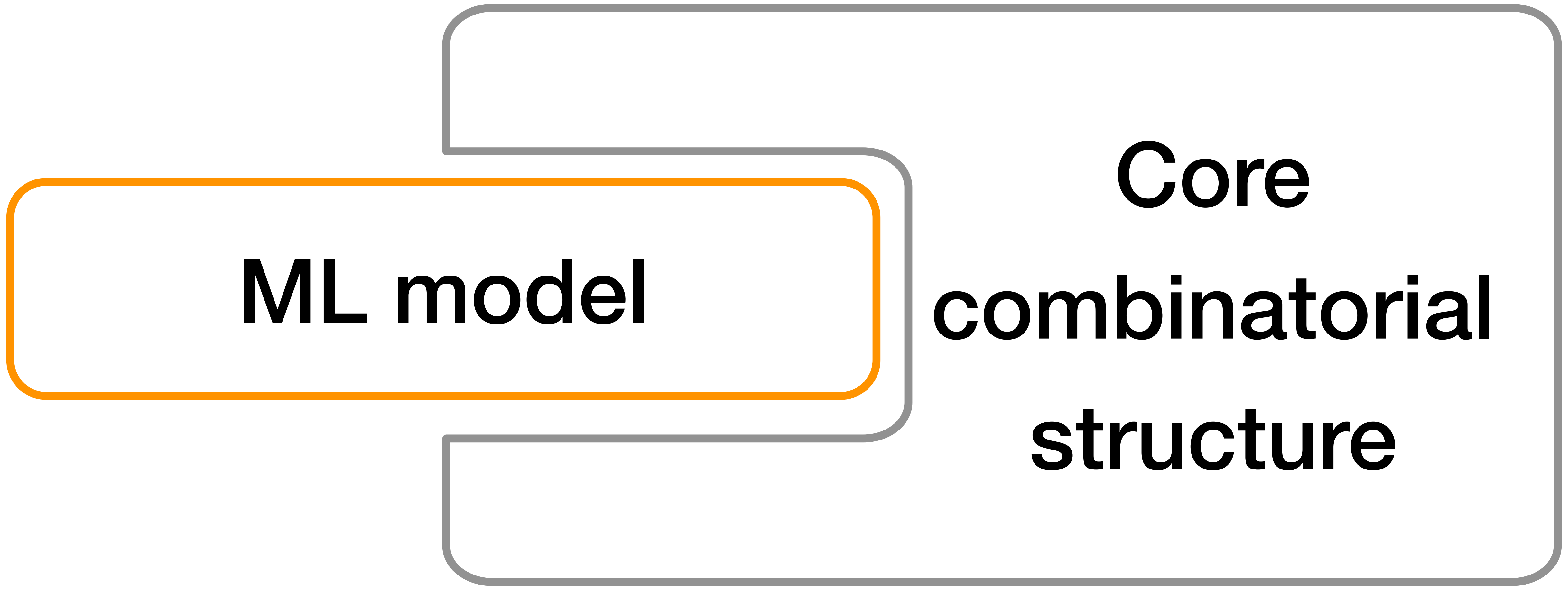
$$\begin{aligned} \min z &= f(\vec{x}, \vec{y}) \\ \text{s.t. } \vec{y} &= h(\vec{x}) \\ &\text{all manner of constraints} \end{aligned}$$

- $\vec{x}$ = ML model input
- $\vec{y}$ = ML model output

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Empirical (Decision) Model Learning

In an nutshell:



EML = combinatorial problem + ML model

Why is it cool?

- Optimize over complex systems!
- Faster then running a simulator
- No simulator, still fine
- Choice of host optimization technology
- Bounding, propagation, inference, etc.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Empirical (Decision) Model Learning

Wait a sec…

EML = combinatorial problem + ML model

Don't we get an <u>approximate</u> model?
😱😱😱

Yes, but:

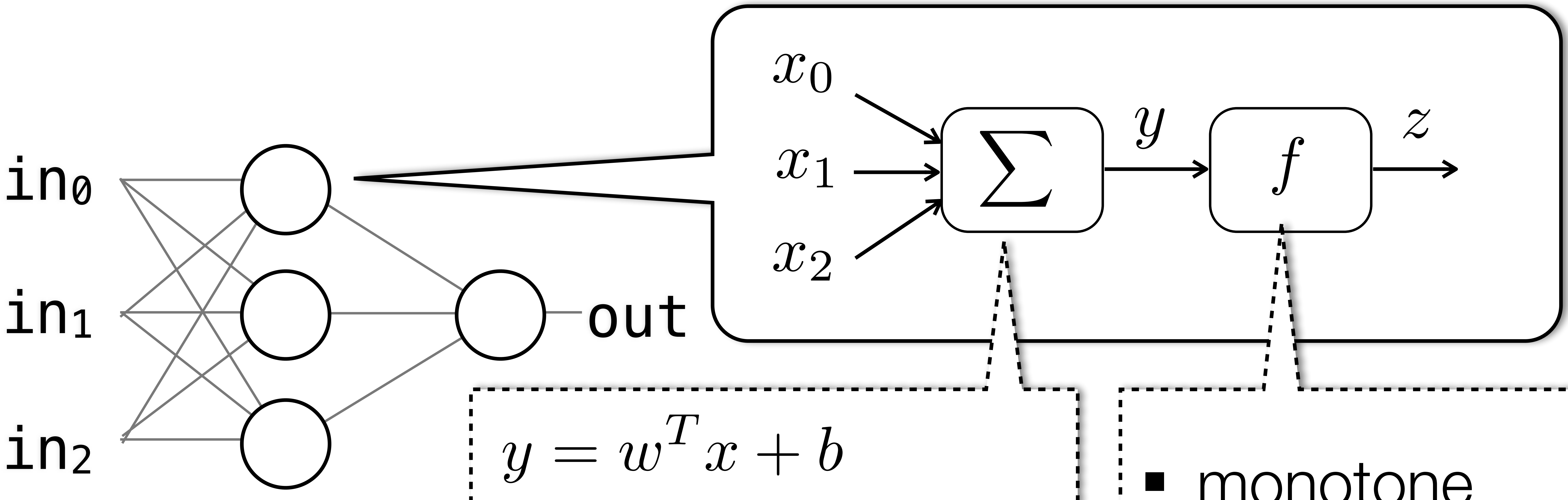
- <u>All models</u> are approximate
- With ML this is acknowledged…
- …And we can even estimate the accuracy!

Key step:

How do we embed a ML model into a combinatorial model?

Neural Networks

Let's consider (Artificial Neural Networks)

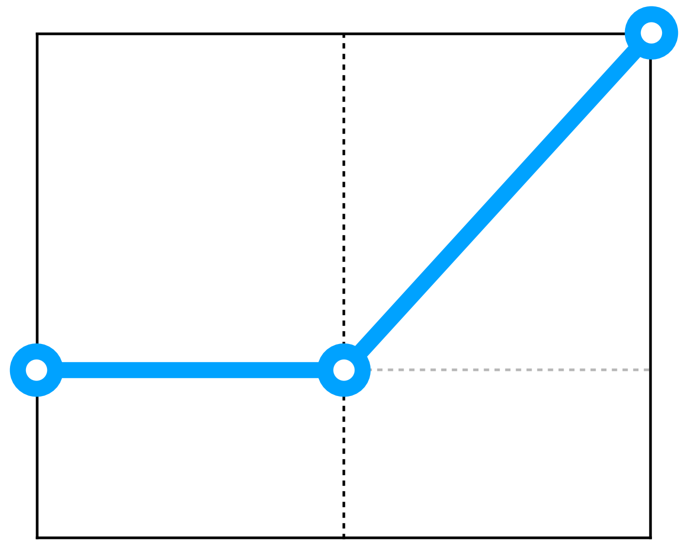


$$y = w^T x + b$$

$$z = f(y)$$

- monotone
- non-decreasing

e.g.
ReLU

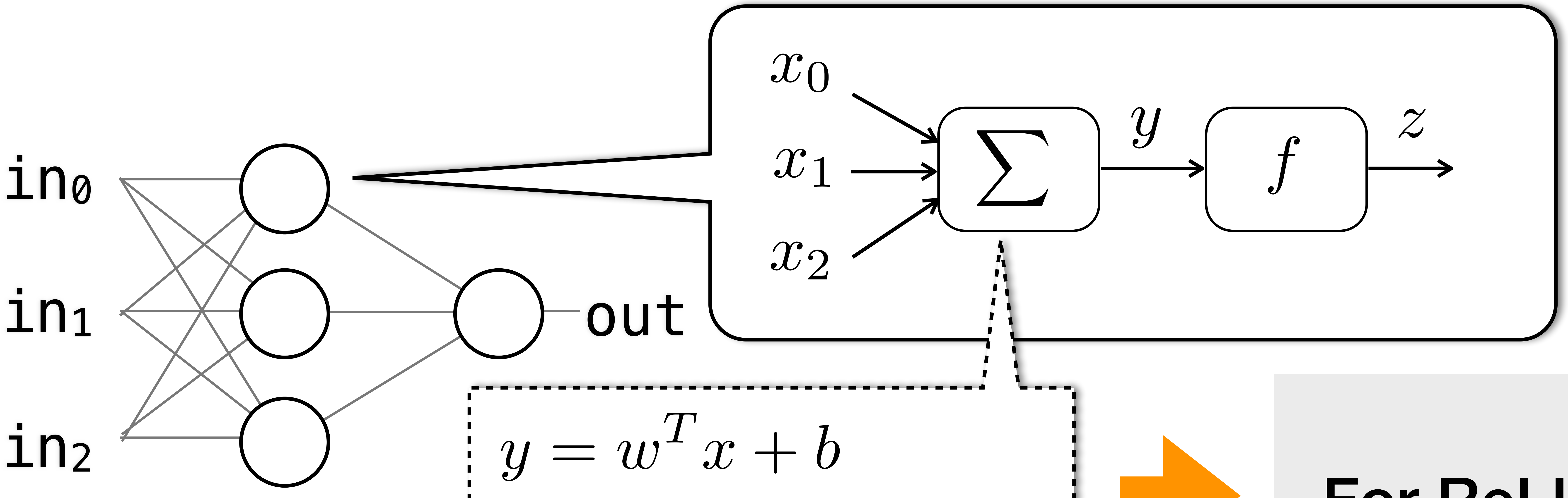


In MI(N)LP: write the NN equations in the model

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Neural Networks & MI(N)LP

Let's consider (Artificial Neural Networks)



$$y = w^T x + b$$
$$z = f(y)$$

For ReLUs:

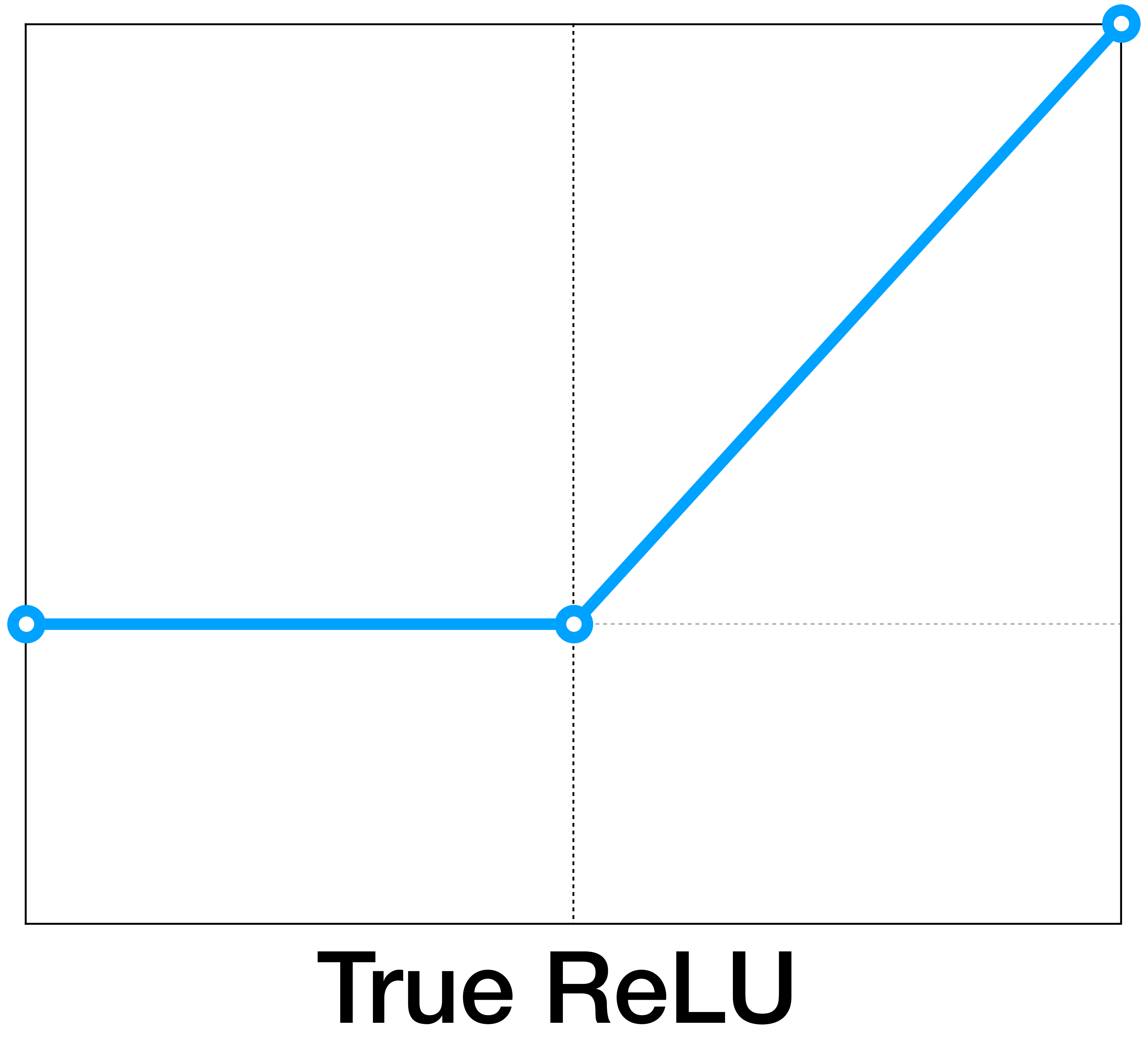
$$y = w^T x + b$$
$$z = \max(0, y)$$

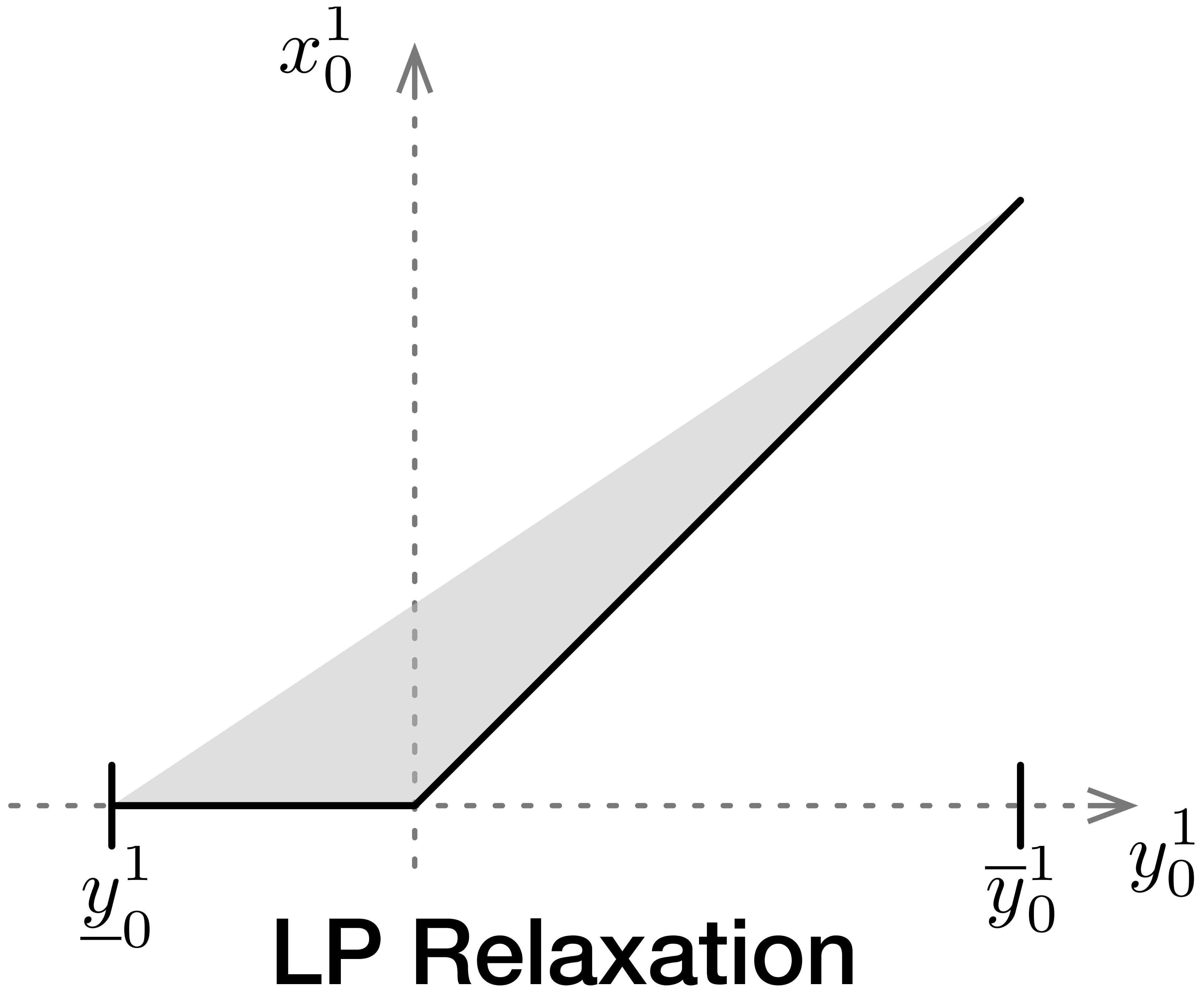
$$z = w^T x + b - s \quad \text{with: } z, s \geq 0$$
$$t = 1 \rightarrow s \leq 0$$
$$t = 0 \rightarrow z \leq 0$$

(indicator constraints)

Neural Networks & MI(N)LP

Sounds simple, but the devil is in the details (i.e. in the bounds):





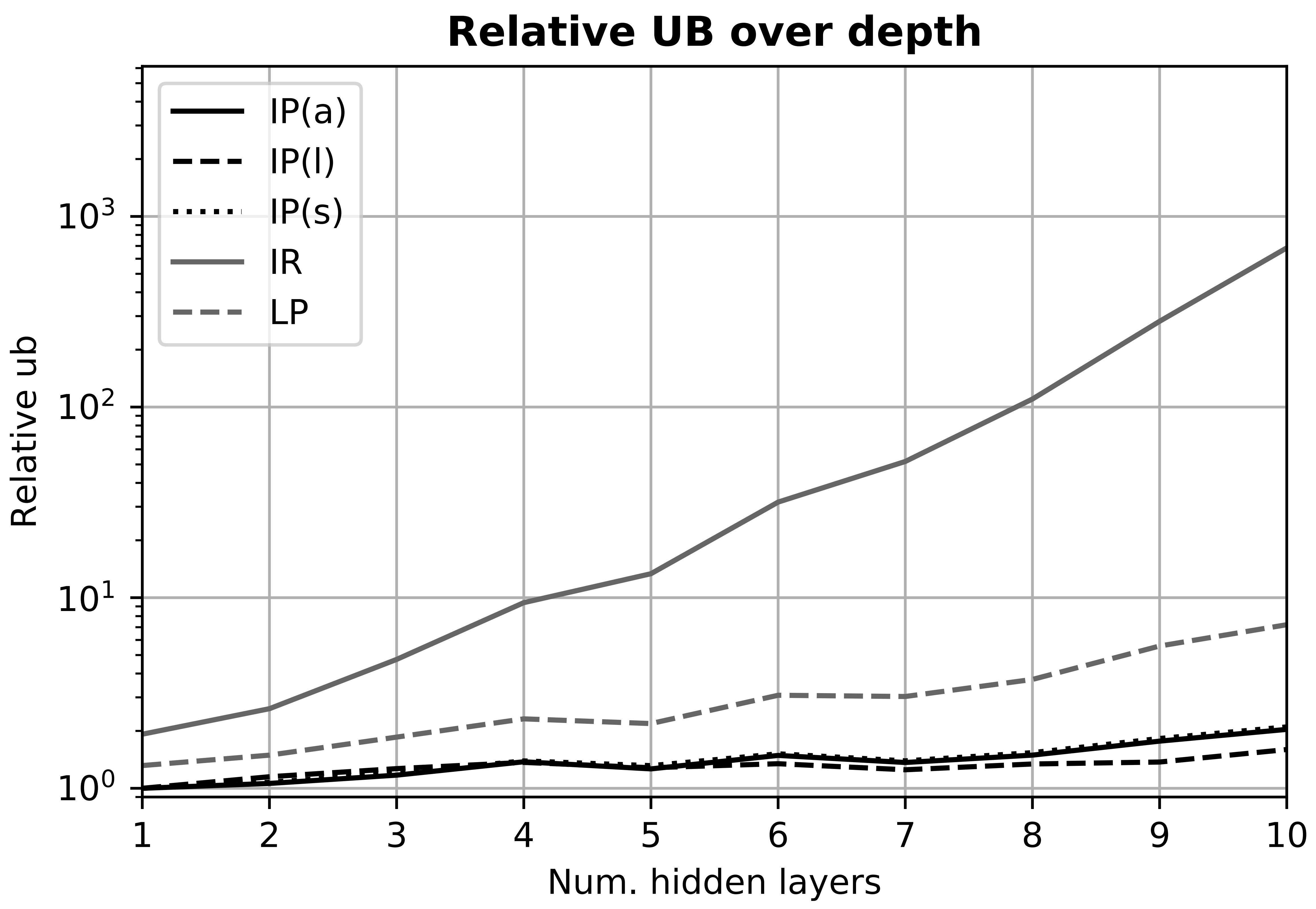
There is a trade-off:

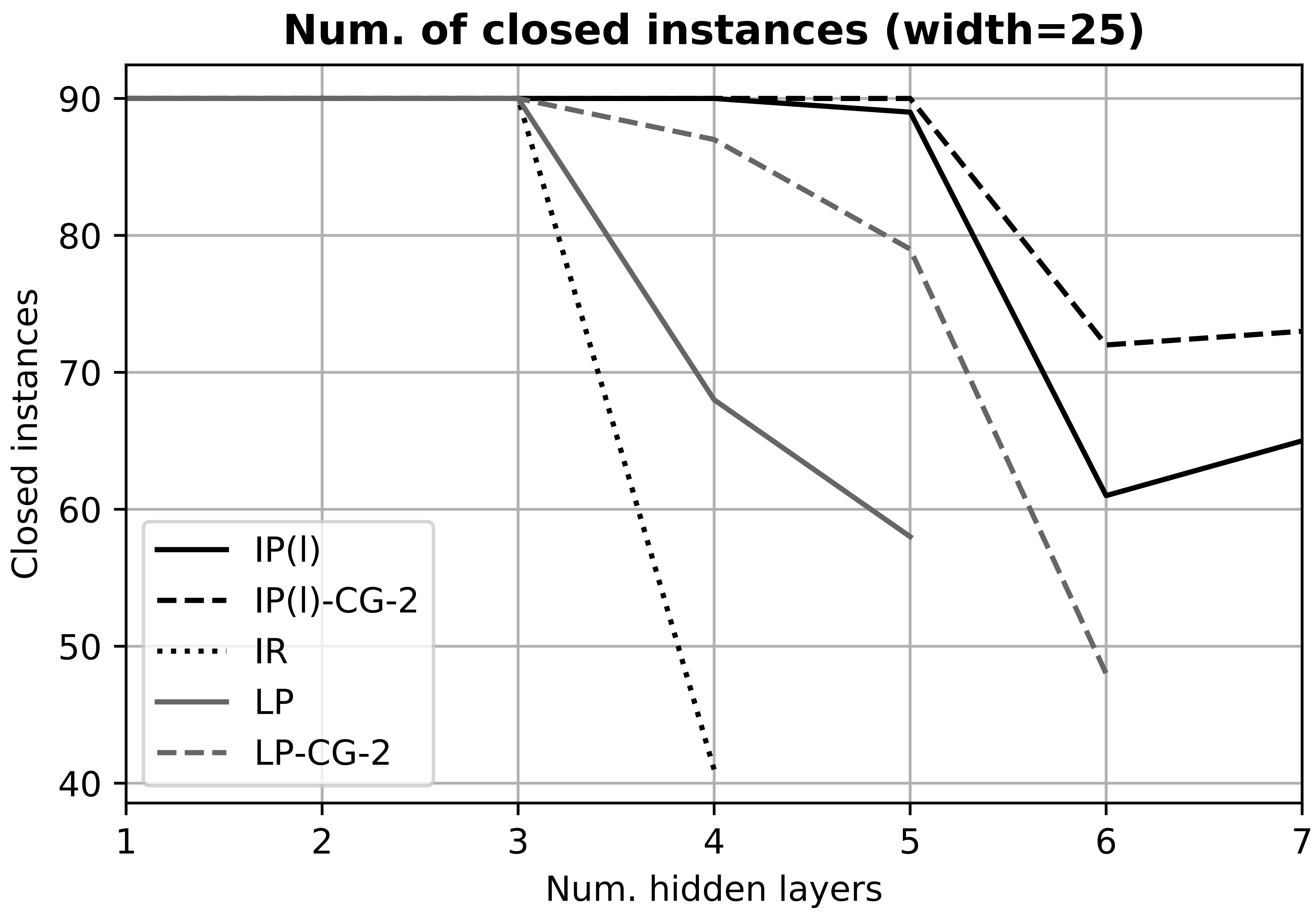
- Poor bounds = poor relaxation
- Good bounds = expensive pre-processing

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Neural Networks & MI(N)LP

Some experimental data:



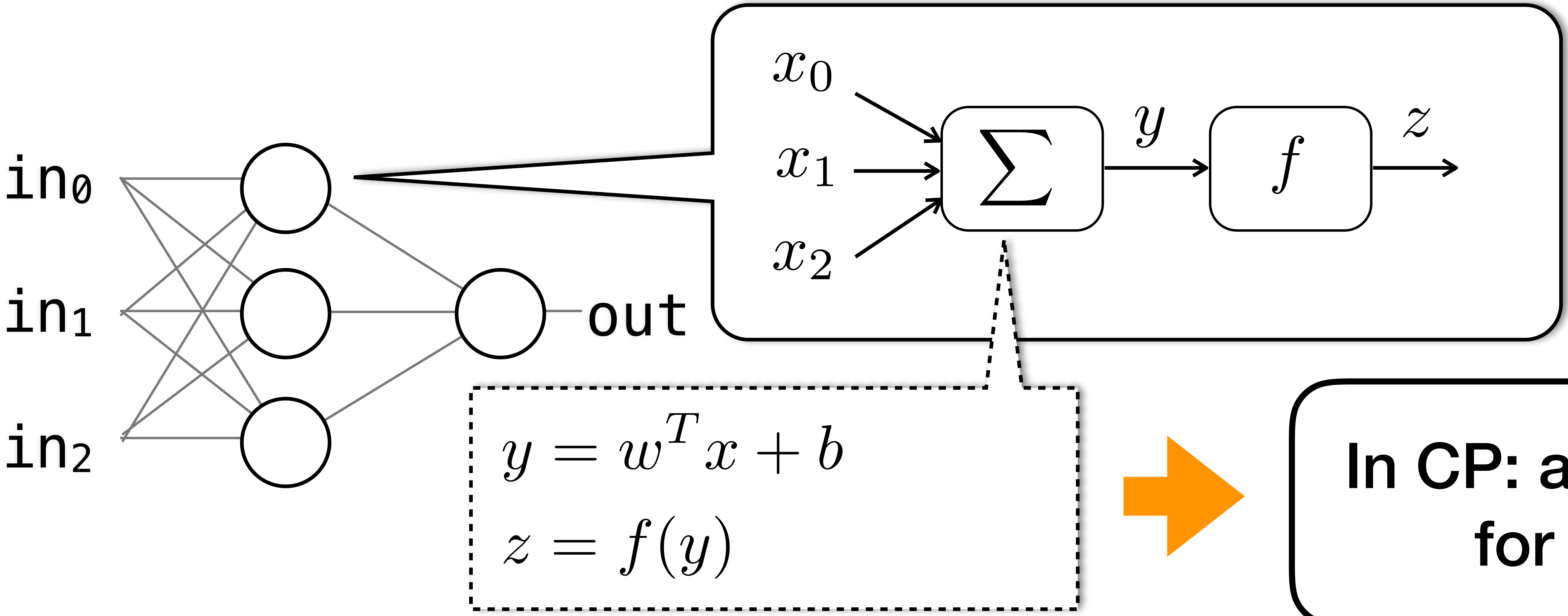


Advice: use strong bound tightening!

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Neural Networks & CP

Let's consider (Artificial Neural Networks)



$$y = w^T x + b$$
$$z = f(y)$$

In CP: a global constraint for each neuron!

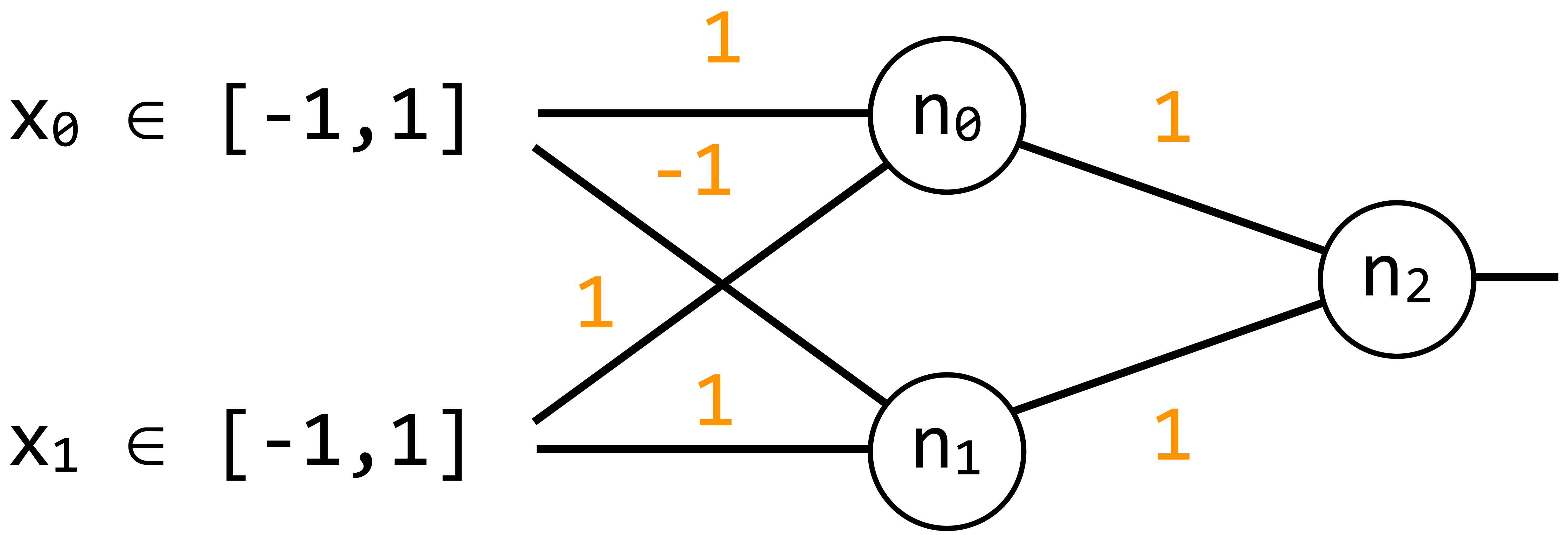
Since *f* is monotone:

- *ub(y)* changes ↔ *ub(z)* changes
- *lb(y)* changes ↔ *lb(z)* changes

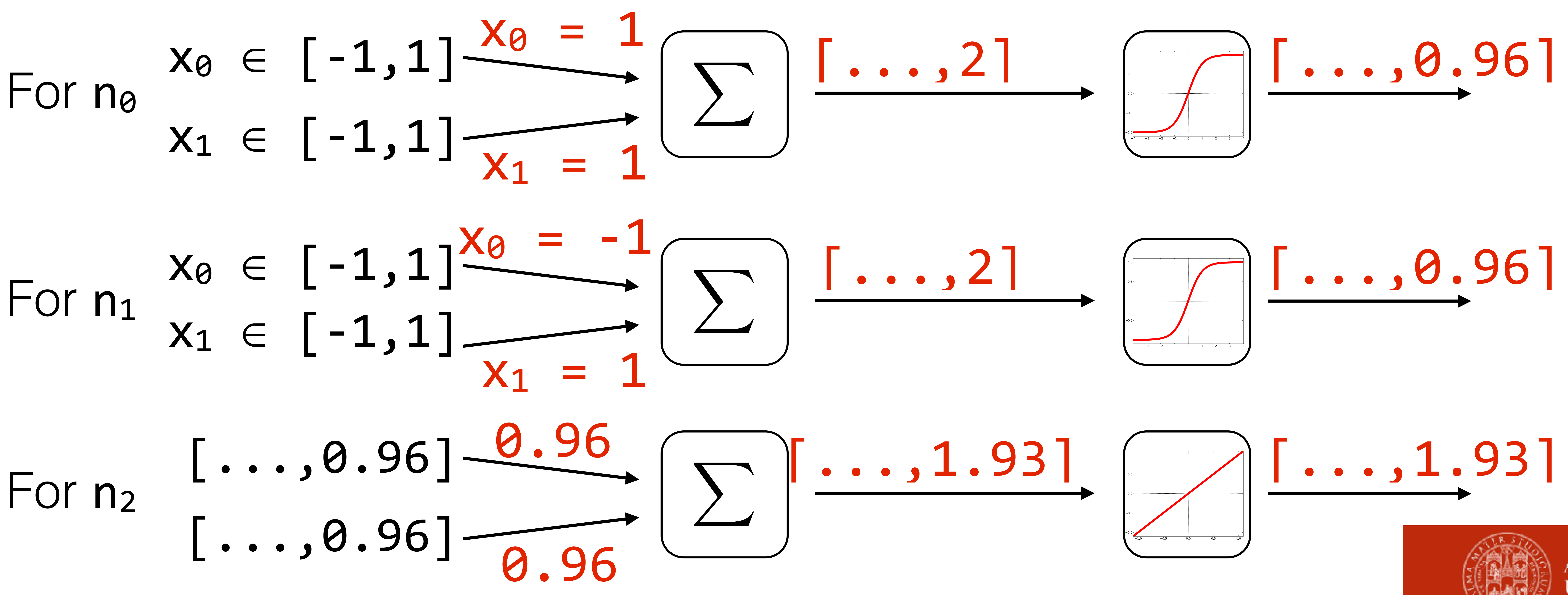
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Neural Networks & CP

However, consider this network:



Propagation:



Neural Networks & CP

The true maximum is 1.51 (not 1.93)!

- There is a discrepancy even for small networks…
- …And it get exponentially worse for large ones

An improvement: Lagrangian relaxation

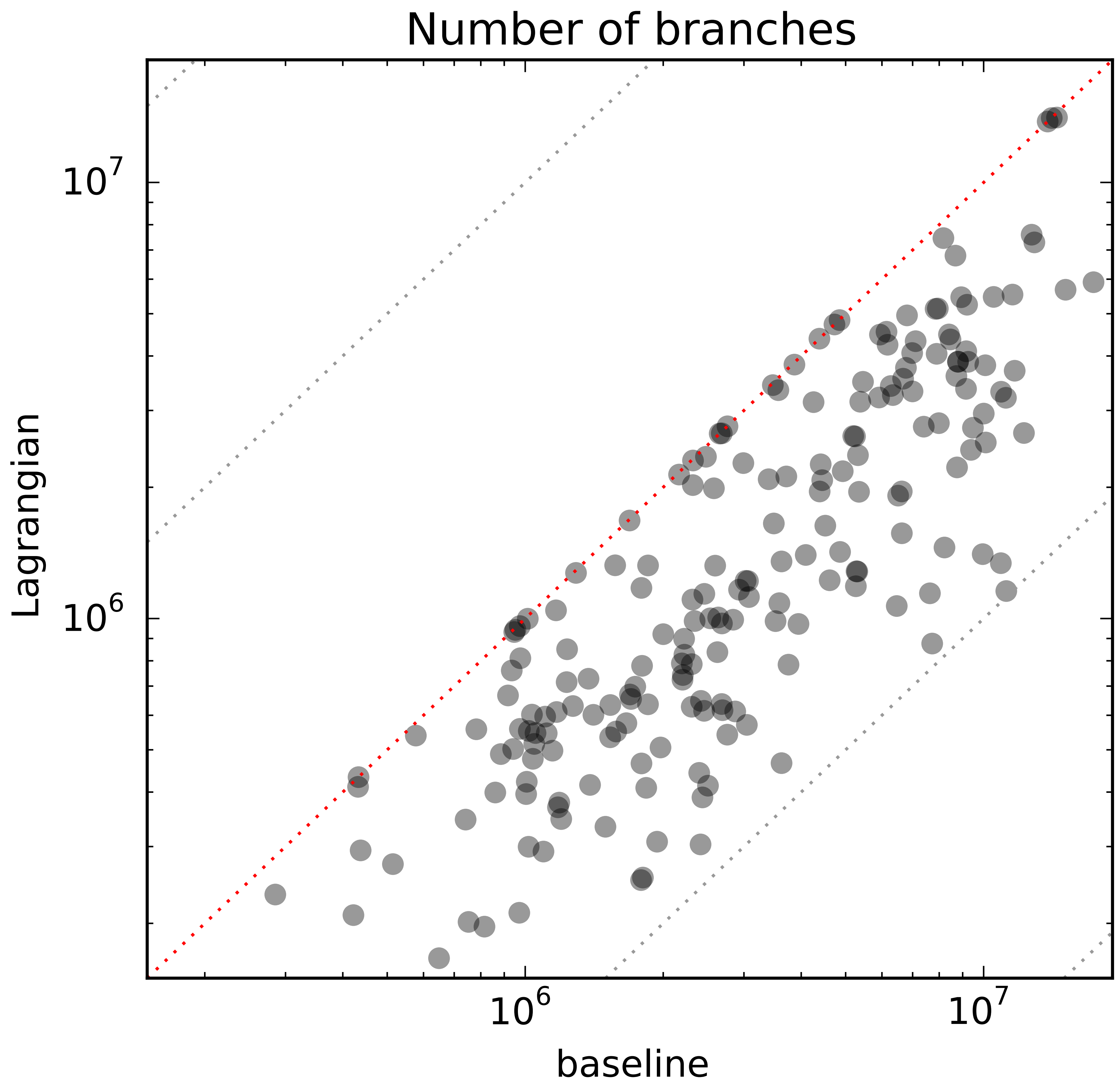
$$\max z(\lambda) = \hat{b} + \sum_{j=0}^{m-1} \hat{w}_j f(y_j) + $$
$$+ \sum_{j=0}^{m-1} \lambda_j \left(b_j + \sum_{i=0}^{n-1} w_{j,i} x_i - y_j \right)$$
$$x_i \in [\underline{x}_i, \overline{x}_i] \qquad \forall i = 0..n-1$$
$$y_j \in [\underline{y}_j, \overline{y}_j] \qquad \forall j = 0..m-1$$

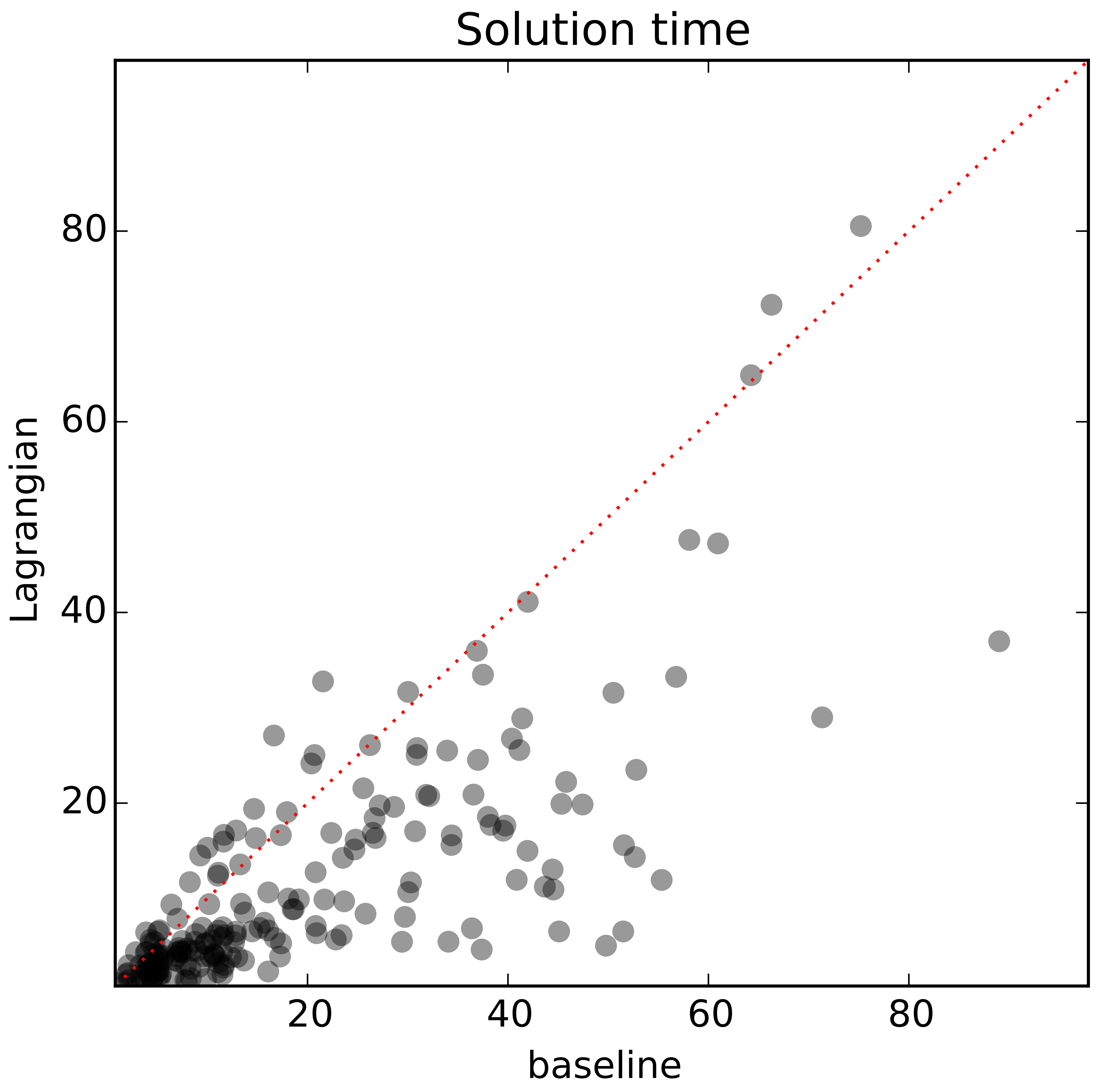
This is separable!

- x-part (linear)
- y-part (non-linear, further separable)

Neural Networks & CP

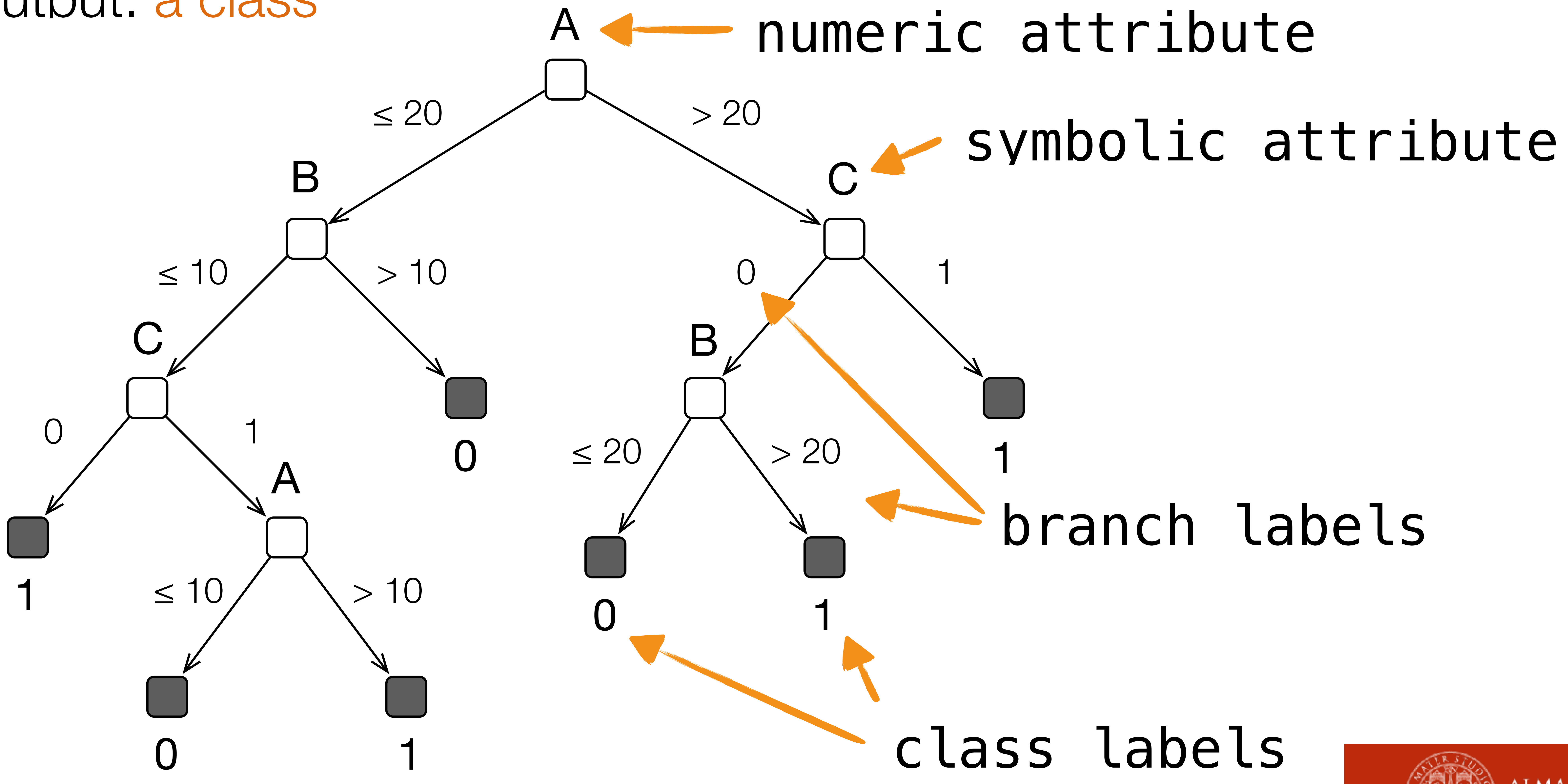
Some experimental results:





Decision Trees

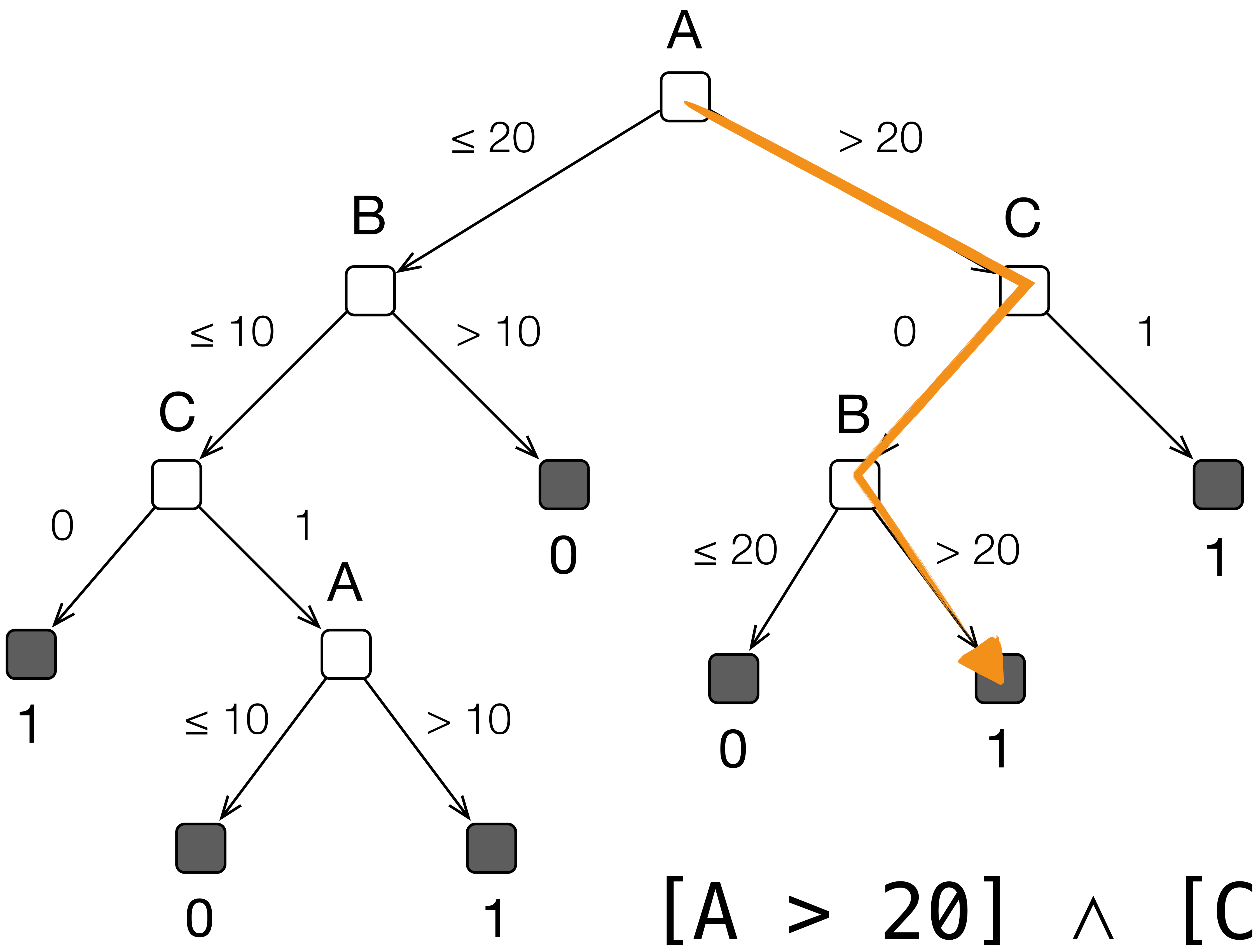
- Input: tuple of attribute values
- Output: a class



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Decision Trees & CP

A first, simple, encoding:



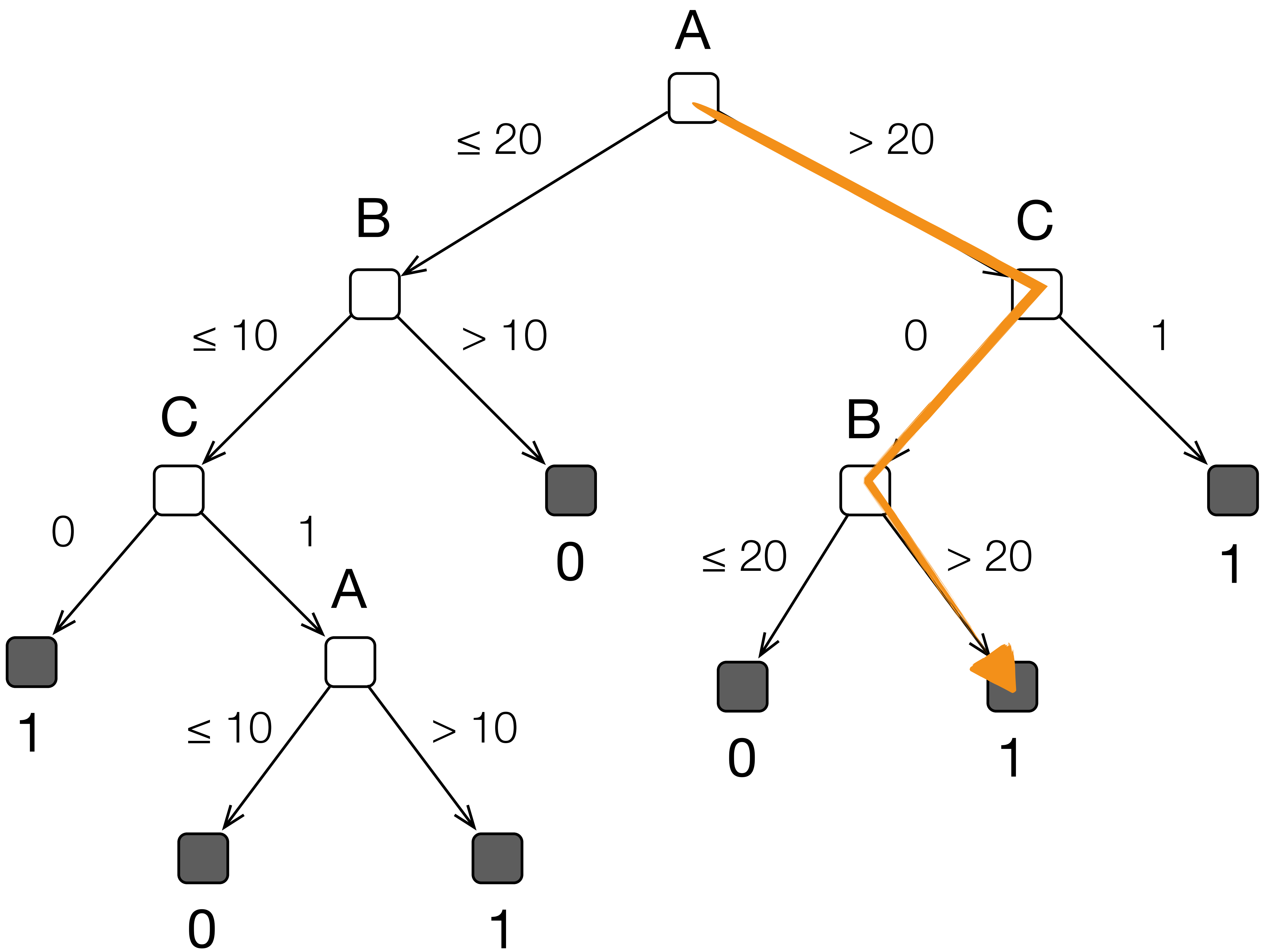
A path is an implication!

$$[A > 20] \land [C = 0] \land [B > 20] \Rightarrow [Y = 1]$$

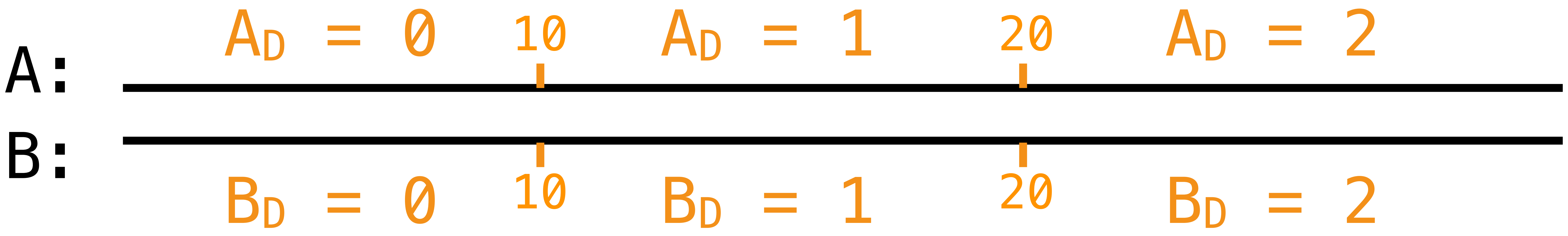
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Decision Trees & CP

A second, stronger, encoding:



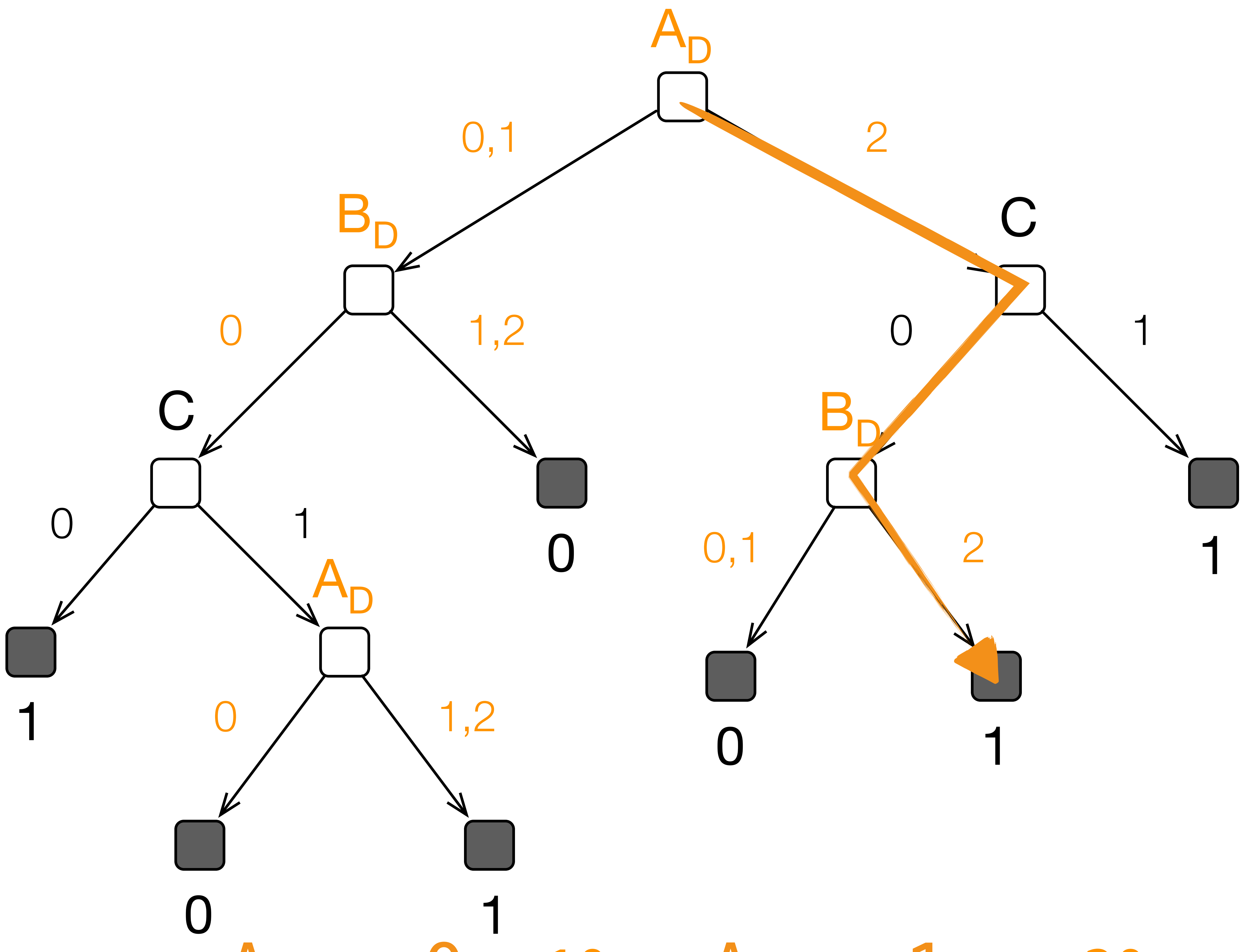
A path is a set of feasible assignments!



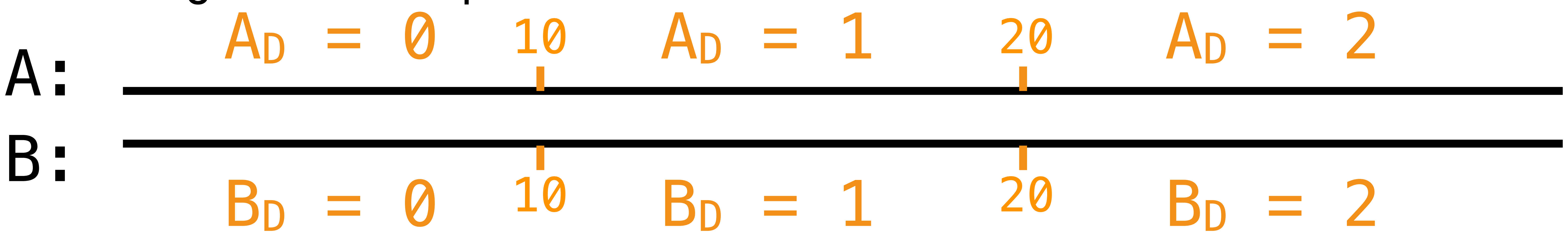
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Decision Trees & CP

A second, stronger, encoding:



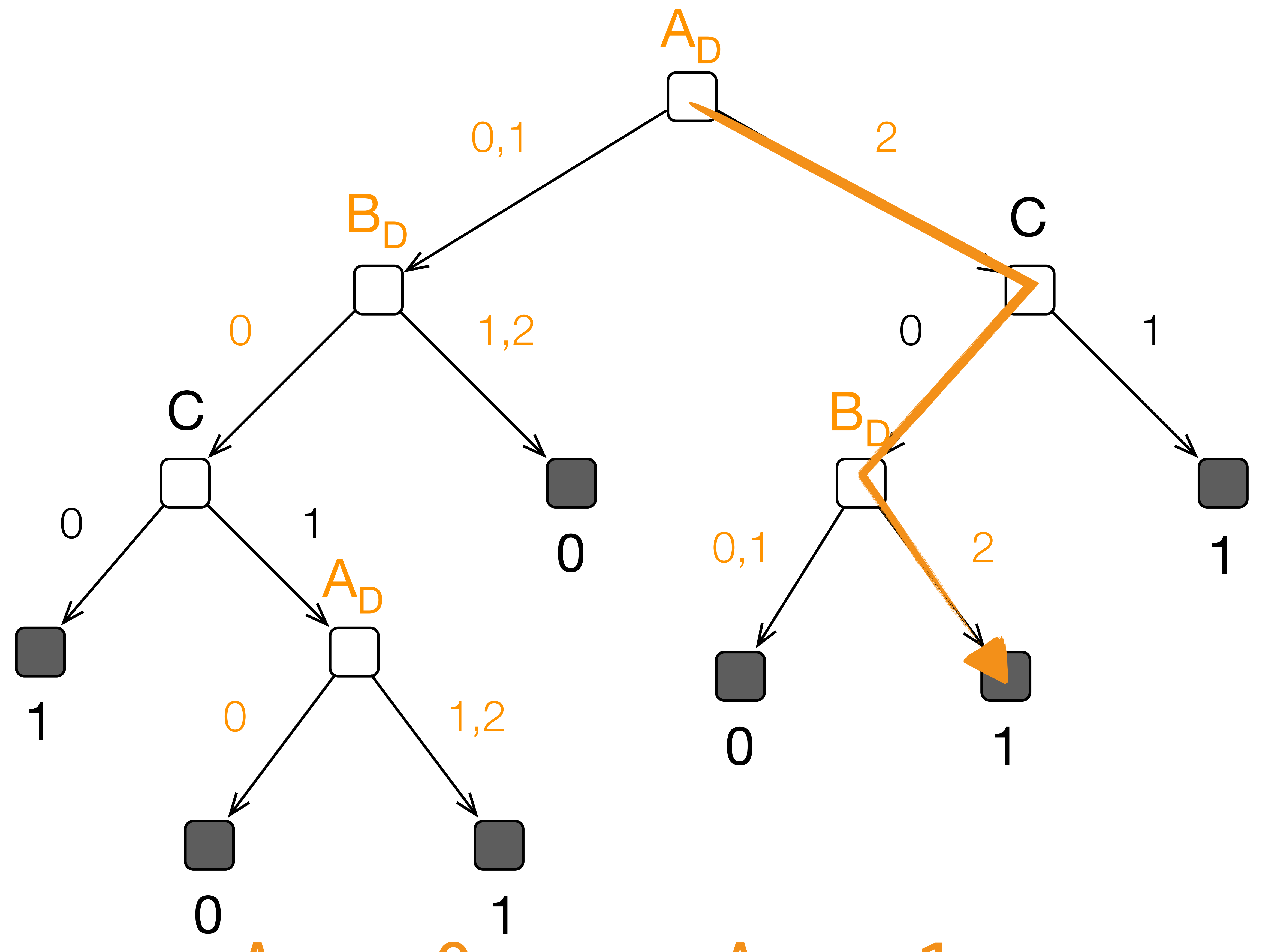
A path is a set of feasible assignments!



ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

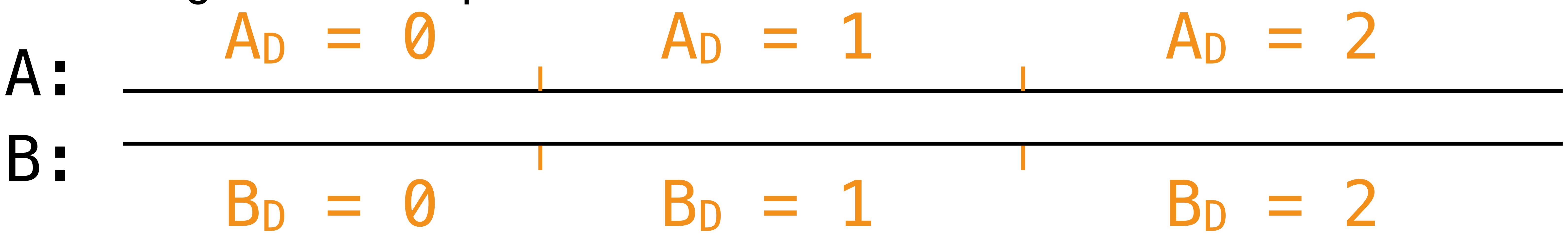
Decision Trees & CP

A second, stronger, encoding:



A path is a set of feasible assignments!

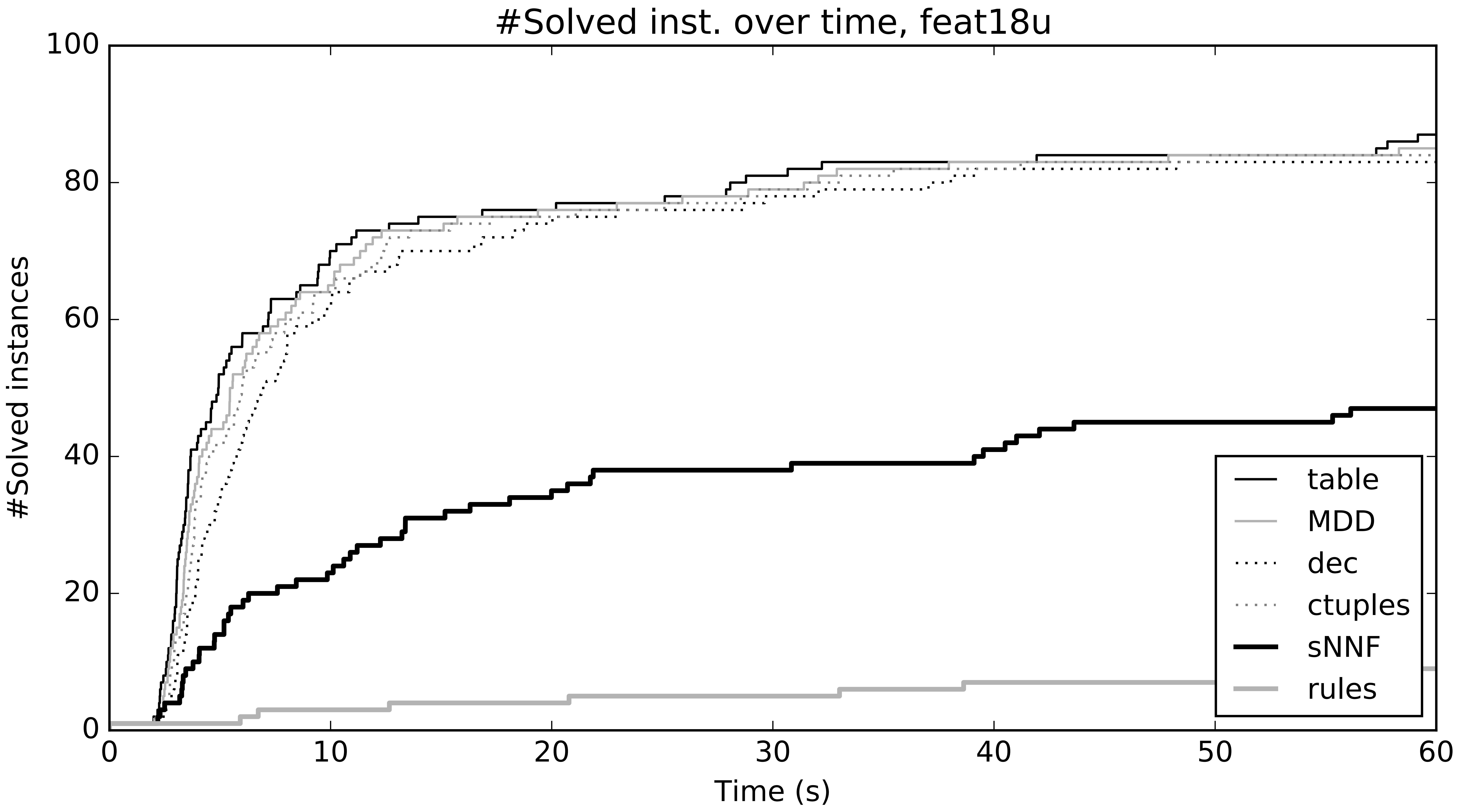
A_D	B_D	C	Y
2	2	0	1
…	…	…	…
…	…	…	…
…	…	…	…



ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Decision Trees & CP

Some experimental results (including other encodings)

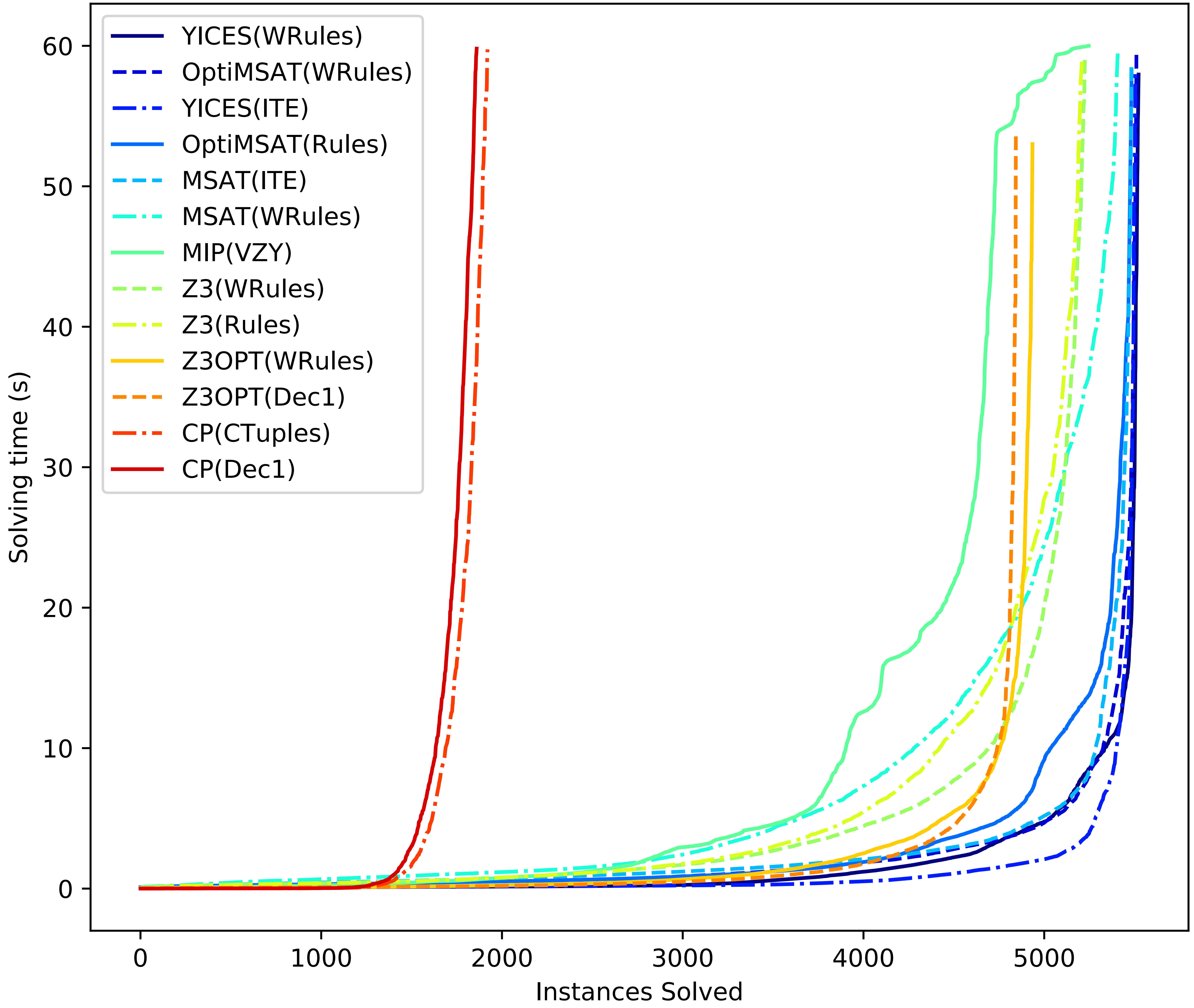


Decision Trees & SMT

What about using SMT?

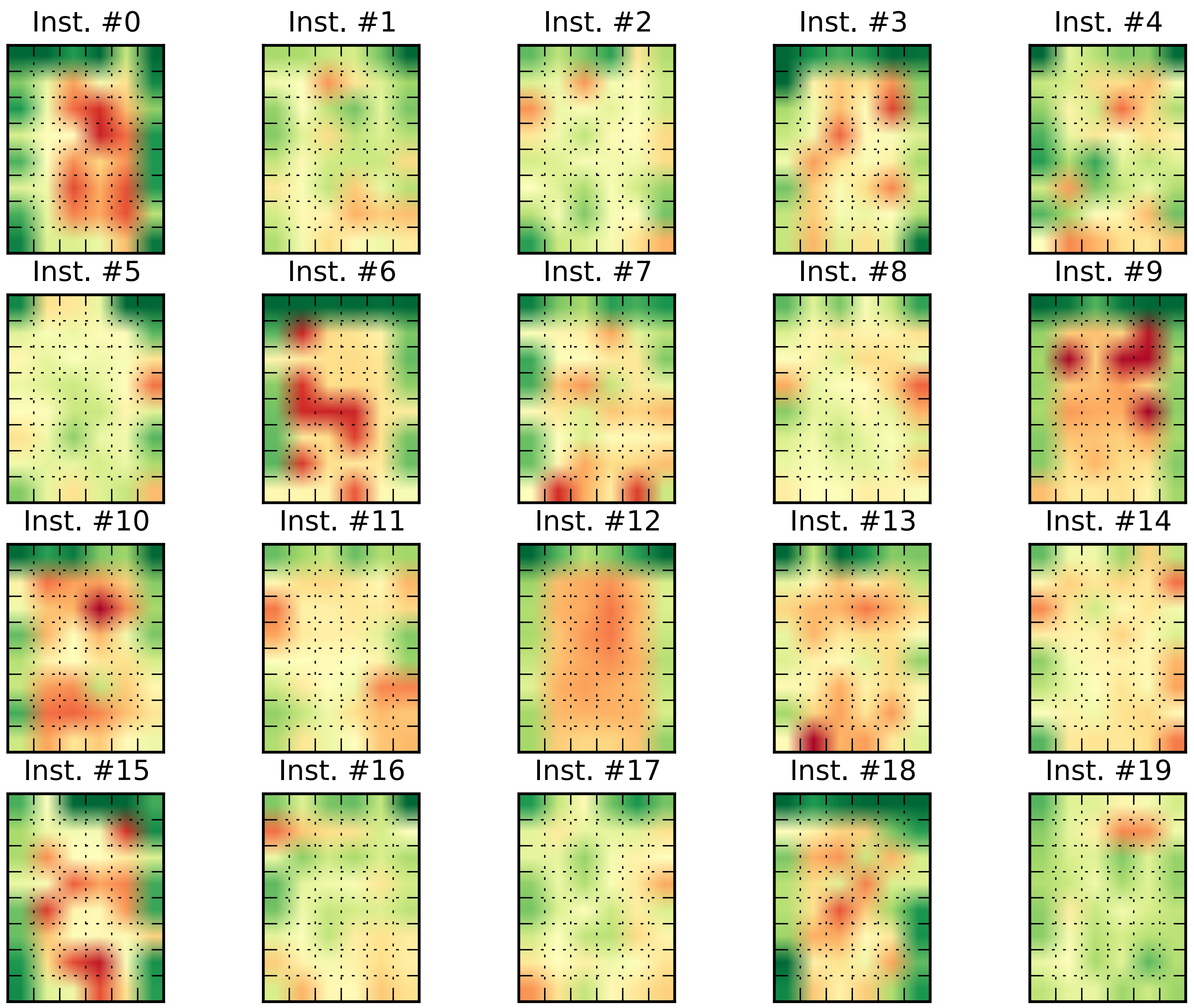
- Easy to encode implications
- No table constraint…
- …But SMT has conflict learning!

Some experimental results:

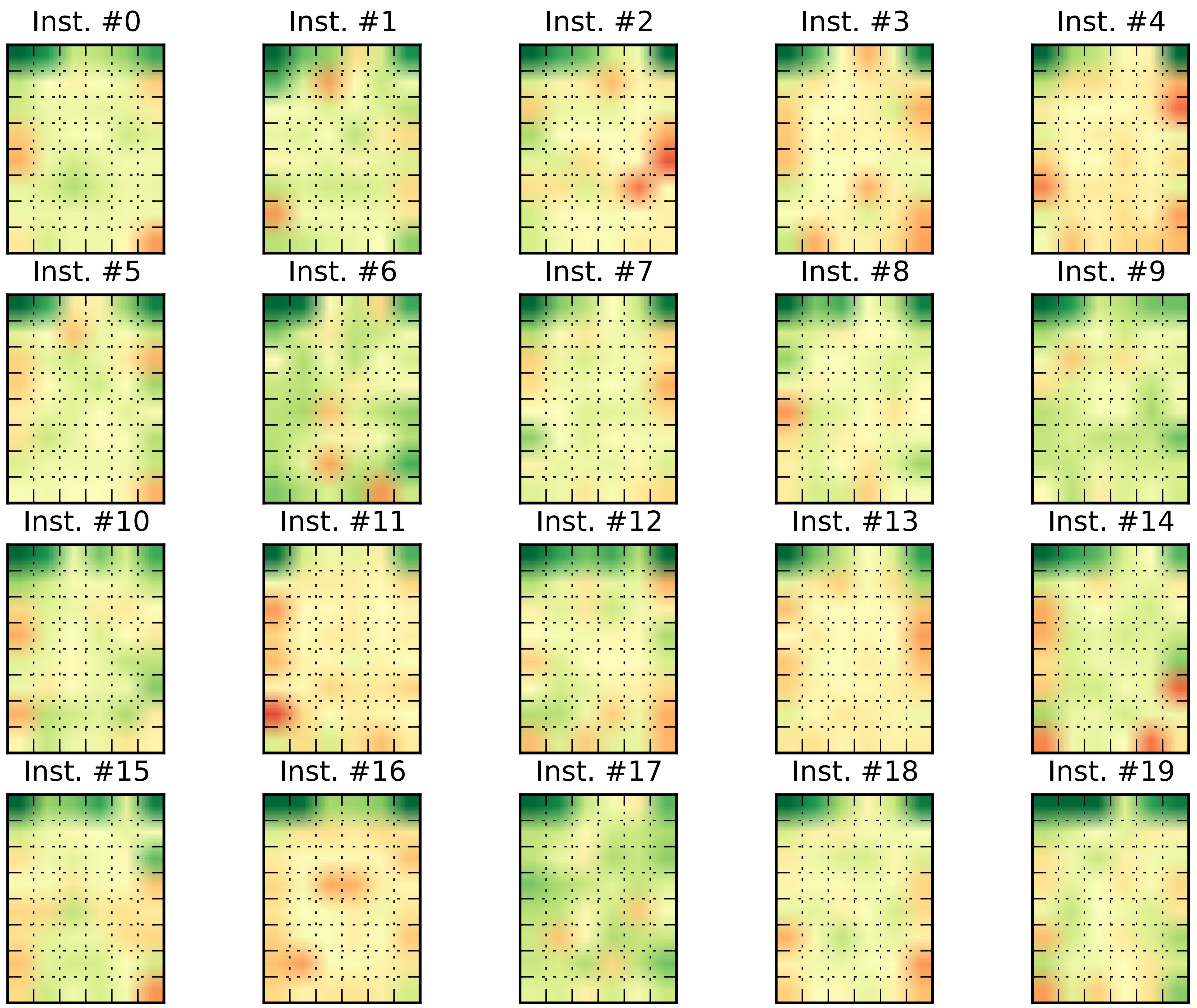


Does it Work? Let's see on the Thermal-Aware Dispatching

True (simulated) core efficiencies, after 60s optimization



Linear Model



NN (ind. neurons) + CP

Af course, the whole picture is bigger…

Of Course, There Are Related Approaches

A bunch of them, in fact:

- Black-box optimization (with surrogate models)
- System identification
- Local search/GAs + actual simulation
- Machine Learning model verification…

We made a survey!

`http://emlopt.github.io`

- A reference web site for all EML-related stuff
- Survey, a (crude) library
- And a decent tutorial with on “epidemic control”…


STOP THE ZOMBIE APOCALYPSE
(with science!)

Food for thought

Morsel #1

EML allows optimization over complex systems

This includes controlled systems!

- The ML can learn the behavior of the system <u>and the optimizer</u>!
- E.g. in thermal aware dispatching we included an <u>on-core scheduler</u>

EML can be used to build hierarchies of optimizers

- CON: no overall optimality guarantee
- PRO: no direct communication, no cuts, etc.!

Morsel #2

In EML, higher accuracy is not always better!

Complex ML models

- More accurate
- Run-time overhead
- Weaker inference (bounds, etc.)

Risk: **poor quality solutions**

Simple ML models

- Less accurate
- Quicker to evaluate
- More effective inference

Risk: **apparently good solutions**

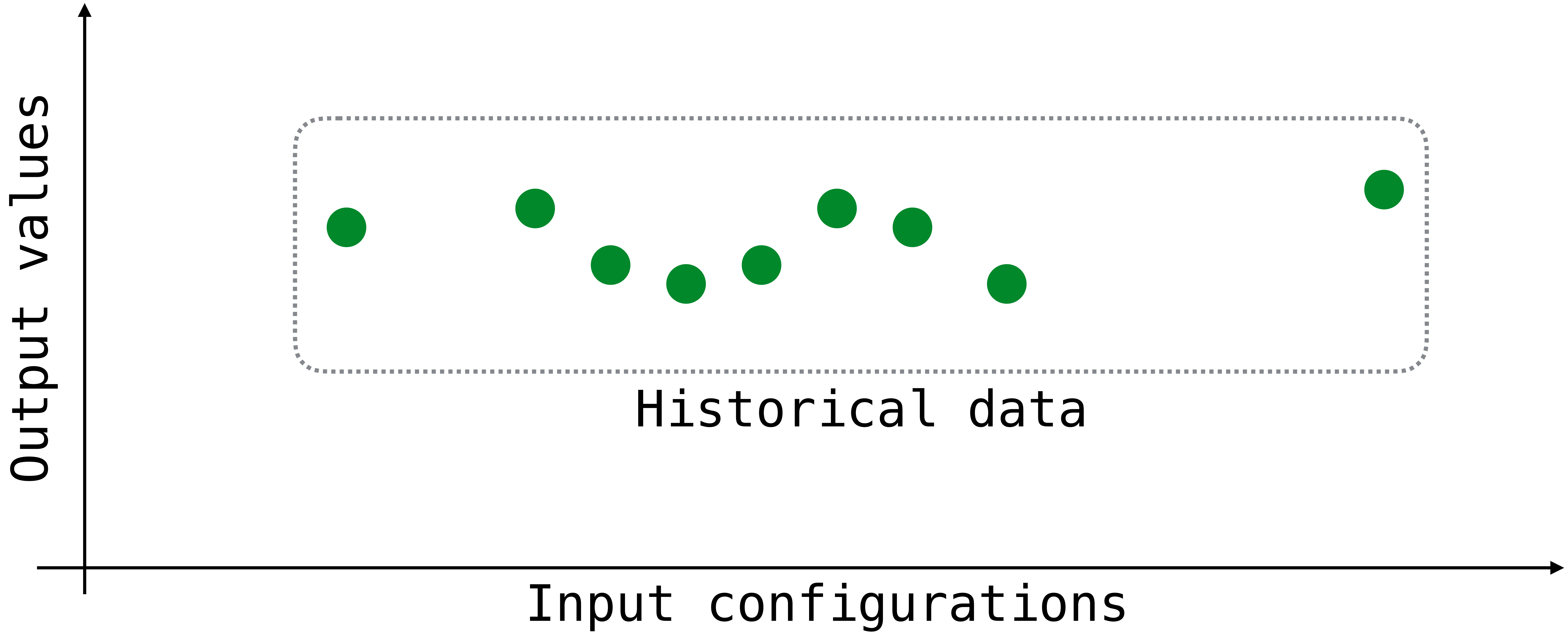
There is trade-off between accuracy and optimization effectiveness

- How to deal with large models?
- How to characterize it?

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

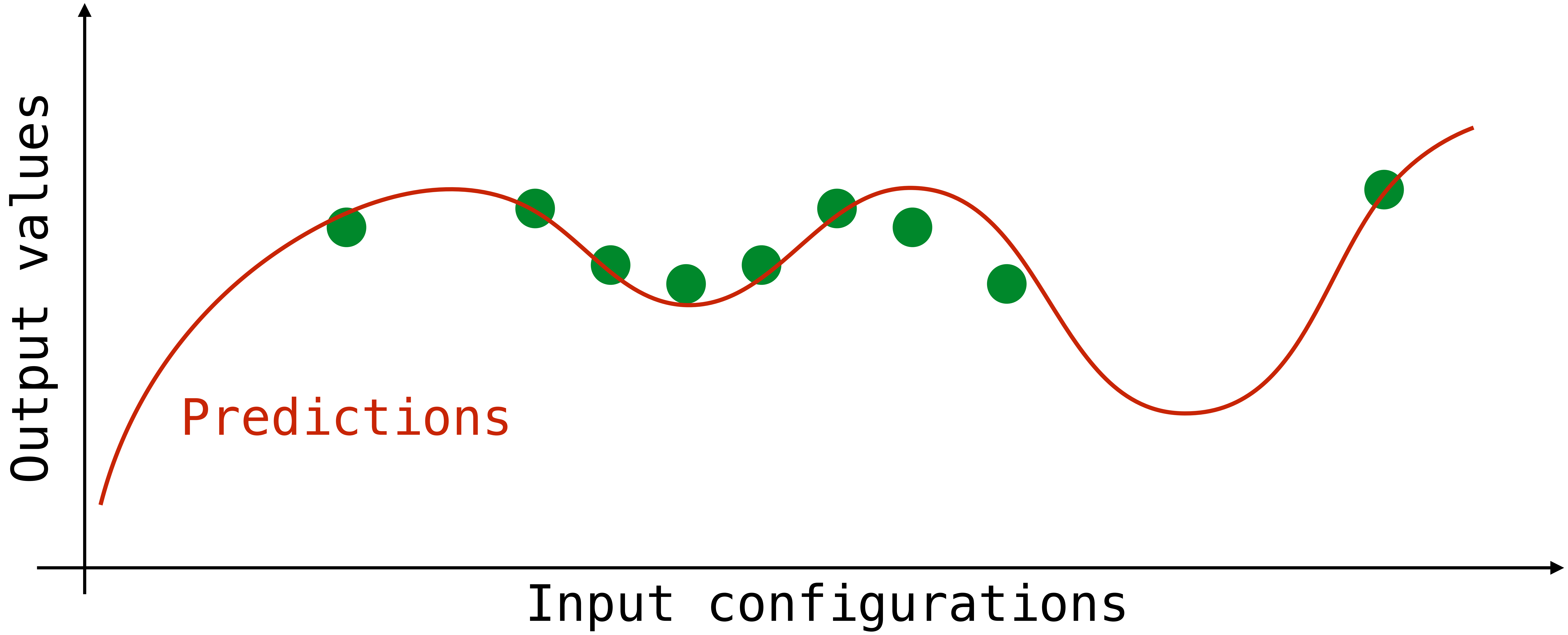
Morsel #3

A typical training set in ML looks like this:



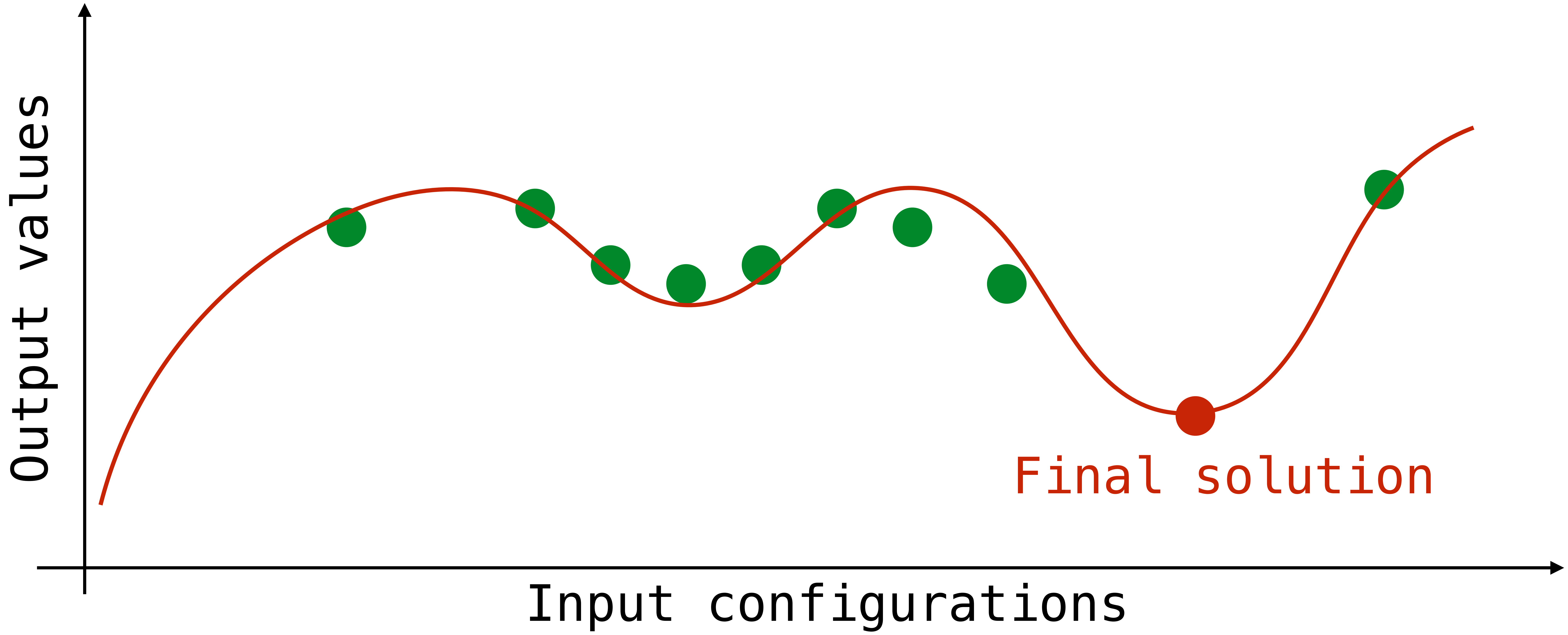
Morsel #3

The ML model provides a prediction for every input value



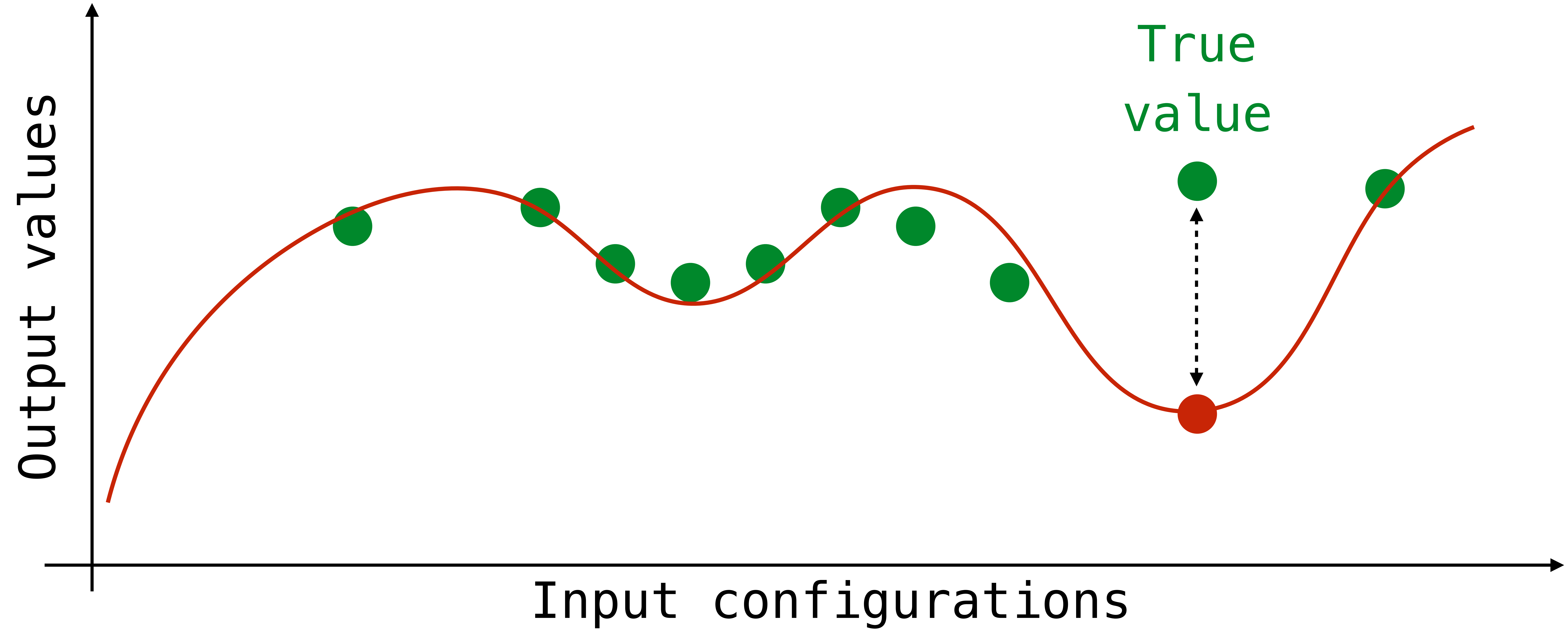
Morsel #3

The optimizer will search for the best one (HP: prediction = cost)



Morsel #3

If this is far from known examples, there may be a large error



ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Morsel #3

What can be done:

- When building the training set
  - Factorial design, Latin hypercube sampling…
- At search time:
  - Active learning, if you can run experiments
  - Connection with preference elicitation and black box optimization

No active learning so far in EML

- Training efficiency?
- How to ensure significant model changes?

Morsel #4

Some ML models provide well-defined uncertain output

- DT report misclassified examples
- Regression trees have standard deviations
- NN classifiers yield full probability distributions

Some ML models can deal with uncertain inputs

- Both DTs and RTs support them nicely

Can we take advantage of this?

- We could do chance constraints via ML!
- Reasoning with stochastic information?

Morsel #5

- Optimization researchers like clean declarative models
- ML researches seldom use decisions as input for their models
- In other fields, simulation and what-if analysis is the way to go

We need to work together!

- Bring together researchers in CP, ML, OR, physics, social sciences…
- Show that optimization on complex real world system is doable!

…And this requires effort from everybody

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

That’s all!
You (also) got questions?

http://emlopt.github.io