ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Empirical Model Learninng

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

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

“CP is well-positioned to pursue the Holy Grail of CS:
the user simply states the problem and the computer solves it”

This is true!

But what problems can we state?

An Example: Traffic Light Placement

- Add/remove traffic lights in a city
- Installation costs and budget limit
- **Objective:** improve traffic flow

“Stating the problem”: That’s a Problem

Let’s try to model that:

$$\min z = f(x)$$

$$\text{s.t.} \sum_{i=1}^{n} c_i x_i \leq b$$

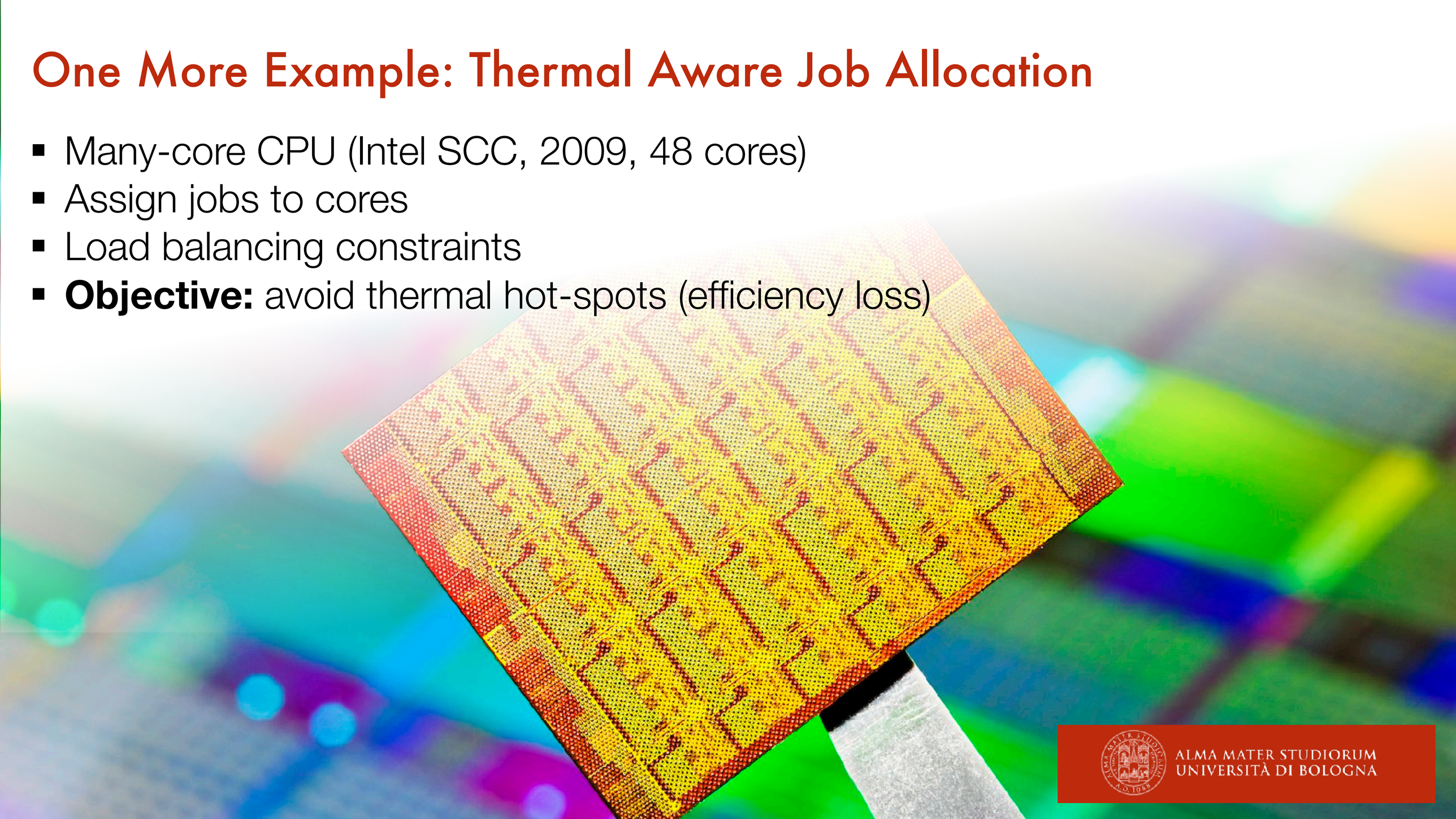
$$x_i \in \{0,1\} \qquad \forall i = 1..n$$

The traffic light impact cannot be modeled by an expert

but we can extract a model from data!

One More Example: Thermal Aware Job Allocation

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



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

One More Example: Trans-Precision Computing

- Tweaking precision levels in computations
- (Stochastic) maximal error guarantees
- **Objective:** minimize energy consumption

What problems can we state?

“Give me a universal solver
and a universal approximator,
and I shall optimize the world”

(Plagiarized Archimedes)

And in theory we have both!

So, what is missing?

Empirical (Decision) Model Learning

- We have universal solvers (CP, SMT, MINLP…)
- We have universal approximators (Neural Nets, DT Ensembles…)

We just need a way to combine the two

Empirical Model Learning in a nutshell

1) Learn → **ML model**

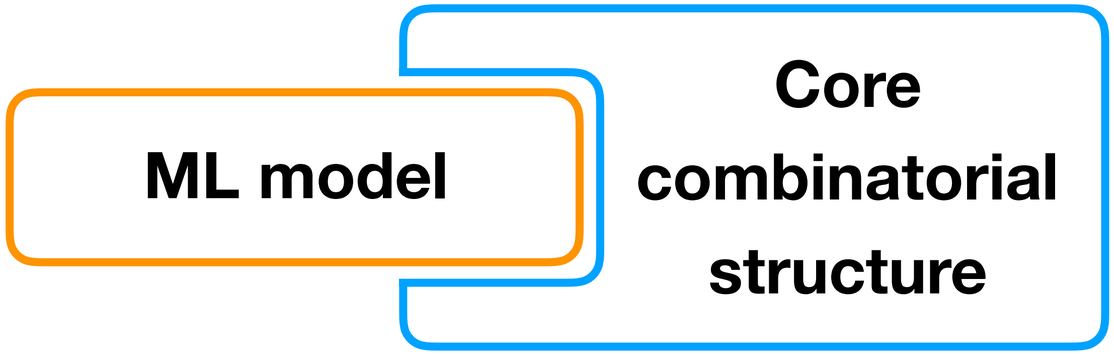
2) Define → **Core combinatorial structure**

Empirical (Decision) Model Learning

- We have universal solvers (CP, SMT, MINLP…)
- We have universal approximators (Neural Nets, DT Ensembles…)

We just need a way to combine the two

Empirical Model Learning in a nutshell



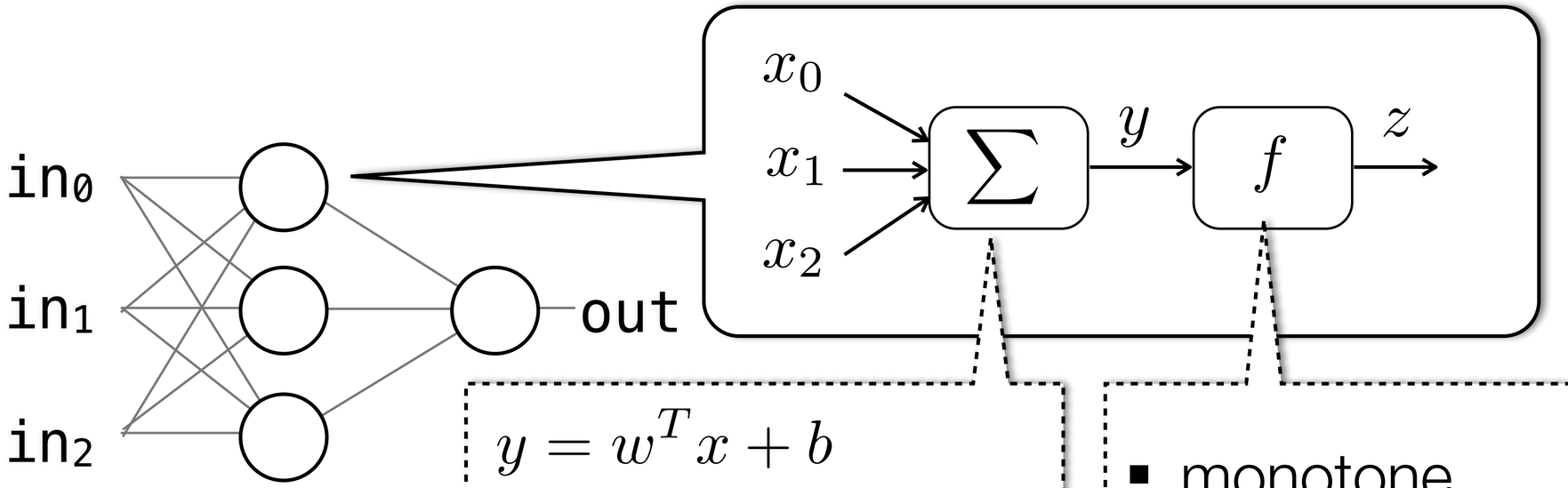
3) Embed!

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

Here come examples on Neural Networks…

Neural Networks & MI(N)LP

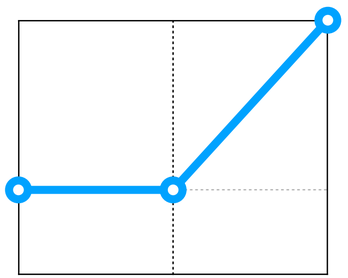
How shall we embed an NN in MI(N)LP?



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

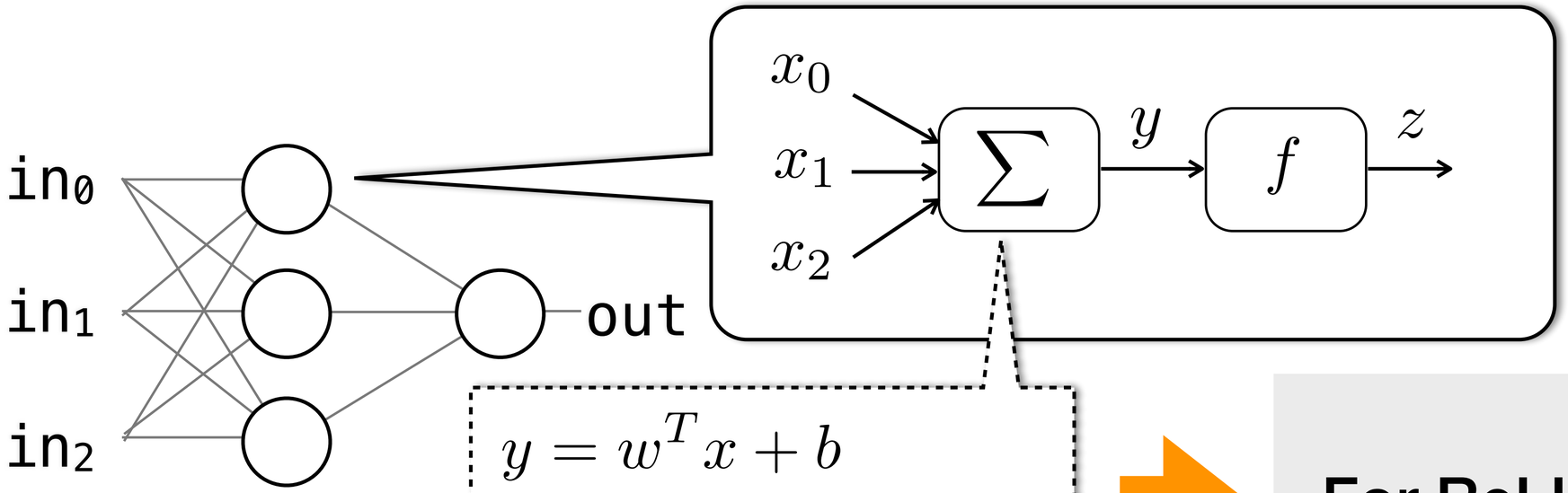
- monotone
- non-decreasing

e.g. ReLU



Neural Networks & MI(N)LP

How shall we embed an NN in MI(N)LP?



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

For ReLUs:

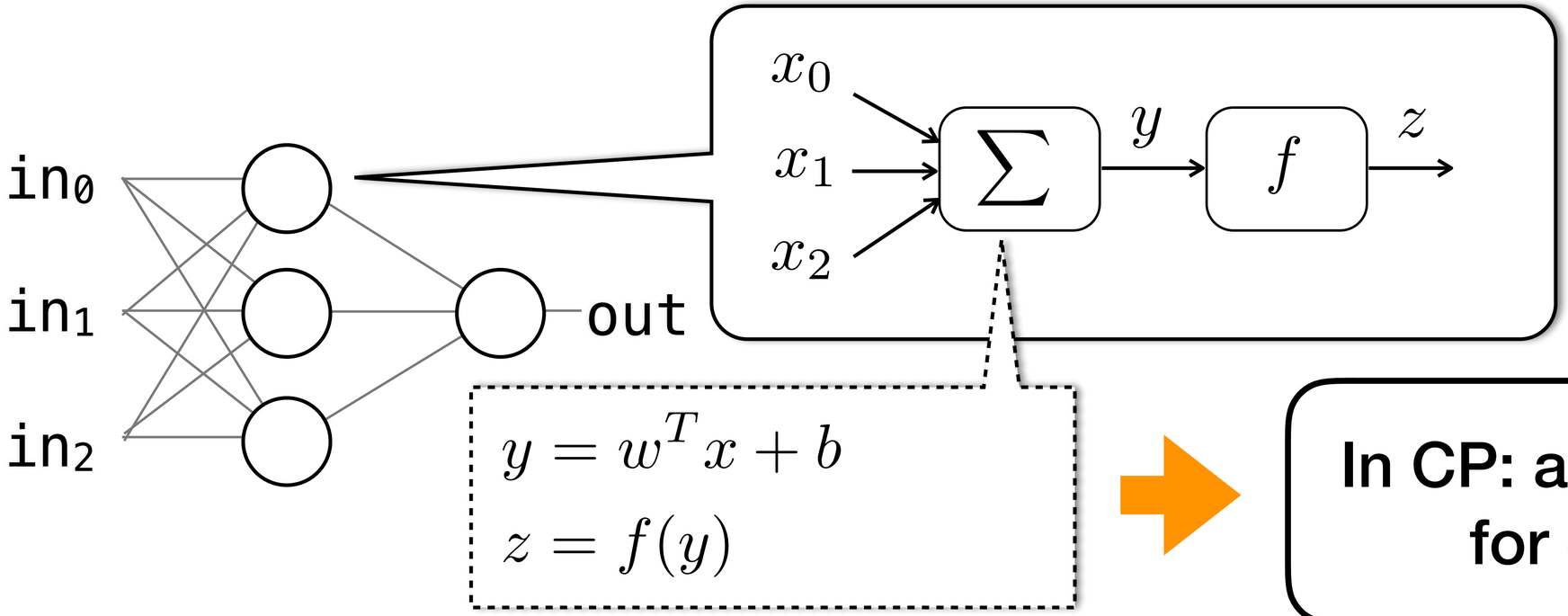
$$y = w^T x + b$$
$$z = \max(0, y)$$

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

(indicator constraints)

Neural Networks & CP

What about CP?



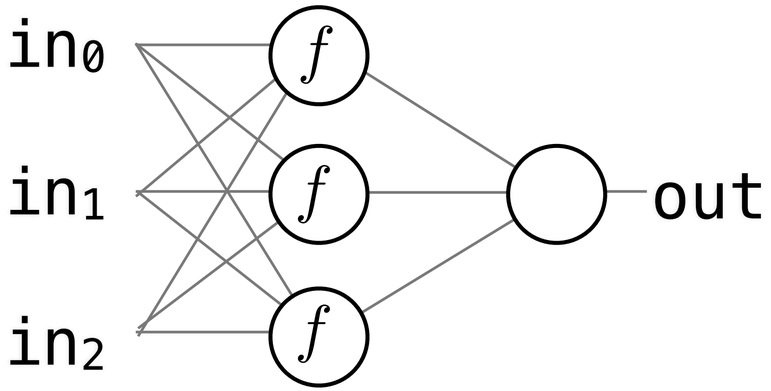
In CP: a global constraint for each neuron!

Since *f* is monotone:

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

Neural Networks & CP

An improvement: 2-layers at once via a Lagrangian relaxation



$$z = \hat{b} + \hat{w} f(y)$$
$$y = Wx + b$$

$$\max z(\lambda) = \hat{b} + \hat{w} f(y) + \lambda^T (b + Wx - y)$$
$$x \in [\underline{x}, \overline{x}]$$
$$y \in [\underline{y}, \overline{y}]$$

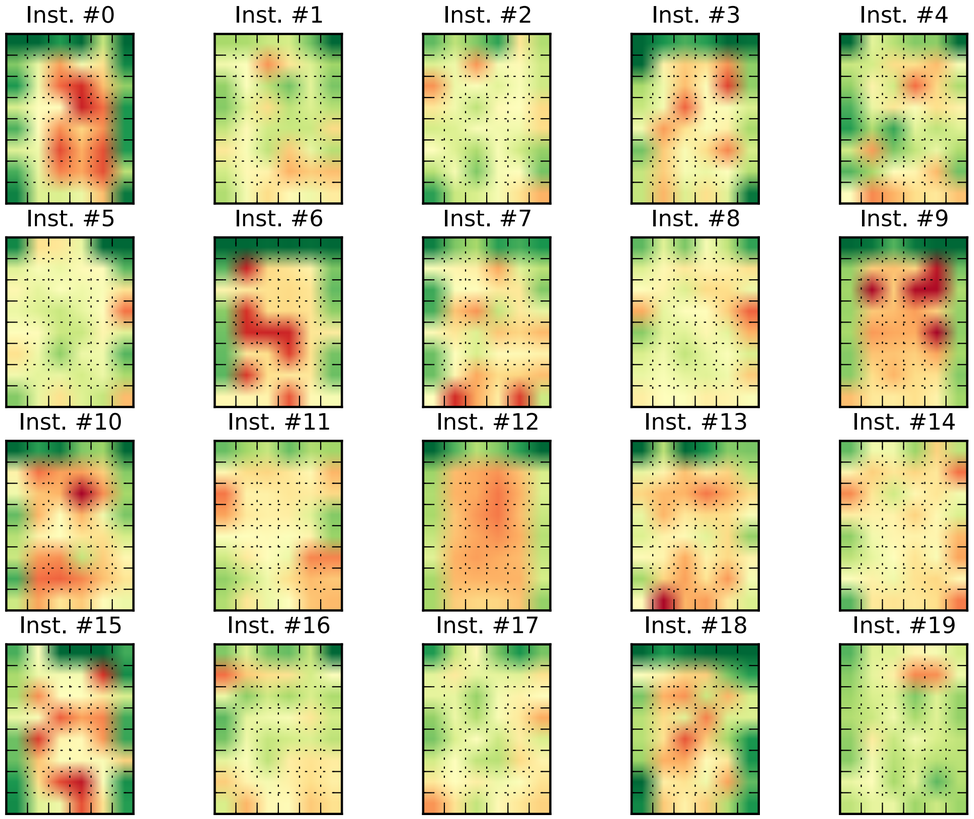
This is separable!

- x-part (linear)
- y-part (again separable)

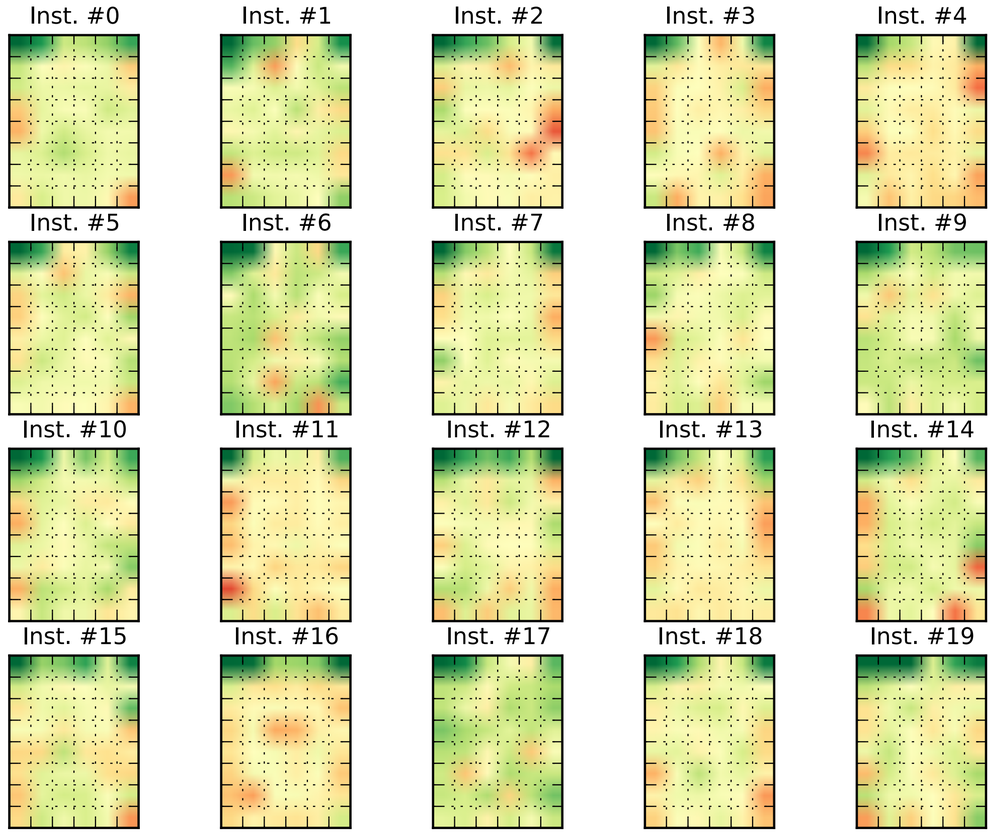
Does it work?

Expert-made Model vs EML

True (simulated) core efficiencies, after 60s optimization



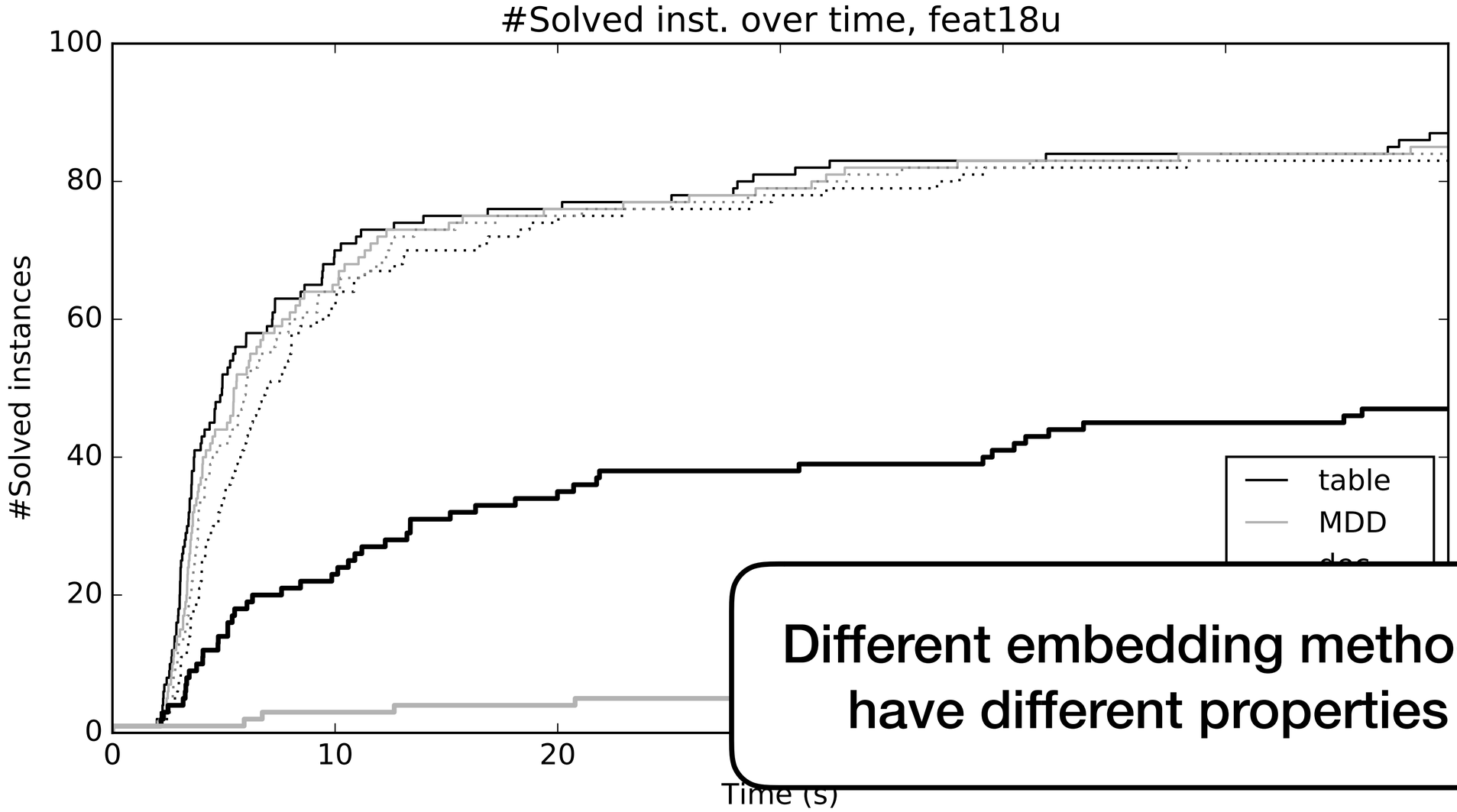
Linear Model



NN (ind. neurons) + CP

Multiple Embedding Methods

Multiple encodings for Decision Trees in CP



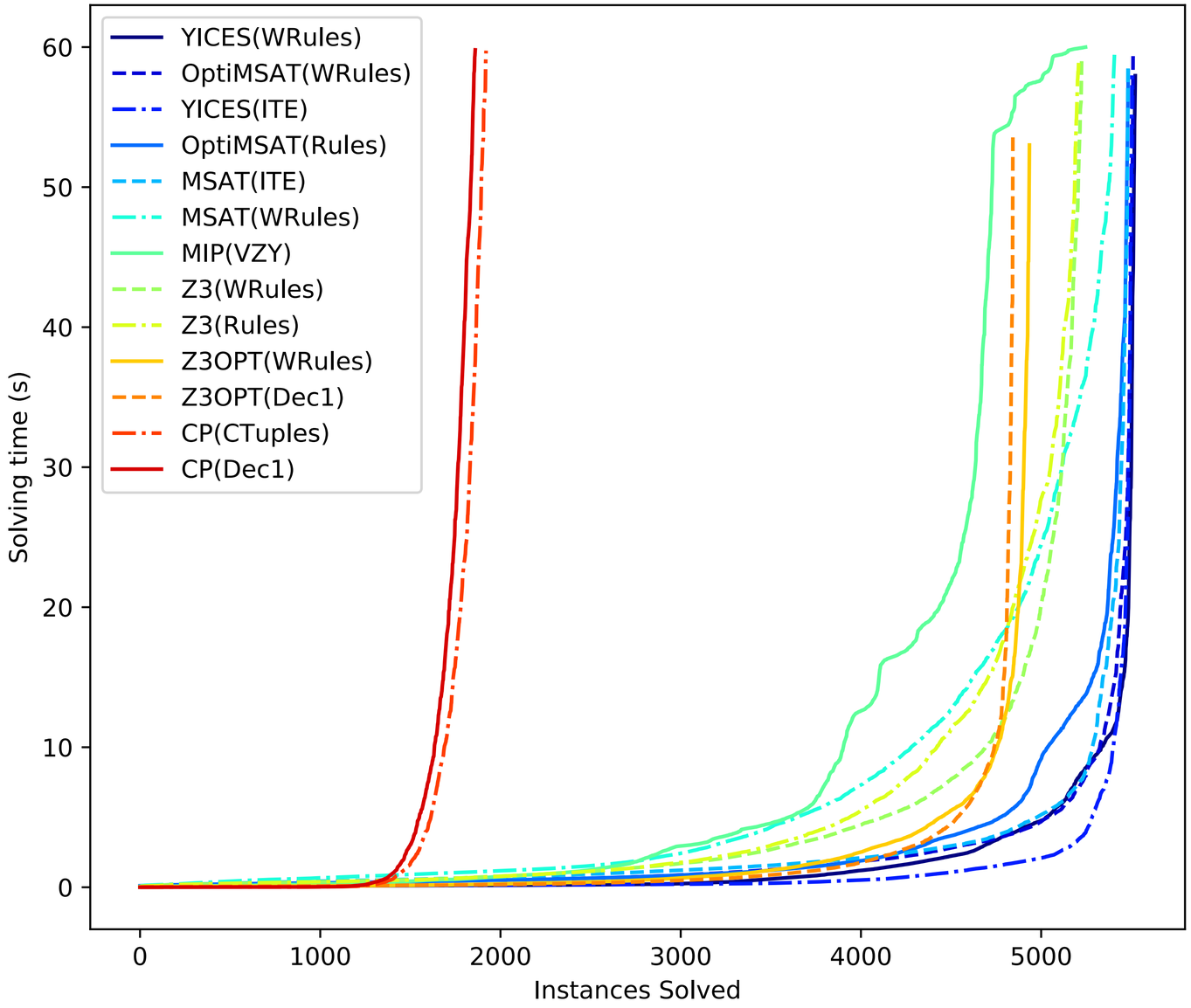
Different embedding methods have different properties

Embedding Methods & Solution Techniques

Best option for Decision Trees

- GAC-capable encodings for CP
- Simple implications for SMT

The solution technique matters (a lot)!



This is only the beginning

How can We Use EML?

Some potential applications

Optimize over hard-to-model systems:

- E.g. job-to-core assignments, traffic lights, equipment set-points

Use EML to model an optimizer and do hierarchical optimization:

- E.g. high-level budget assignment, then low-level budget assignment

Deal with uncertainty:

- E.g. Learn probability distributions
- E.g. a chance constraint as a classifier

ML specific tasks:

- E.g. Verification, counter-examples
- E.g. Explanation

In theory we can optimize the world, but…

“In theory there is no difference
between theory and practice.
In practice, there is”

(A wise person)

Size Does Matter

We have a scalability problem:

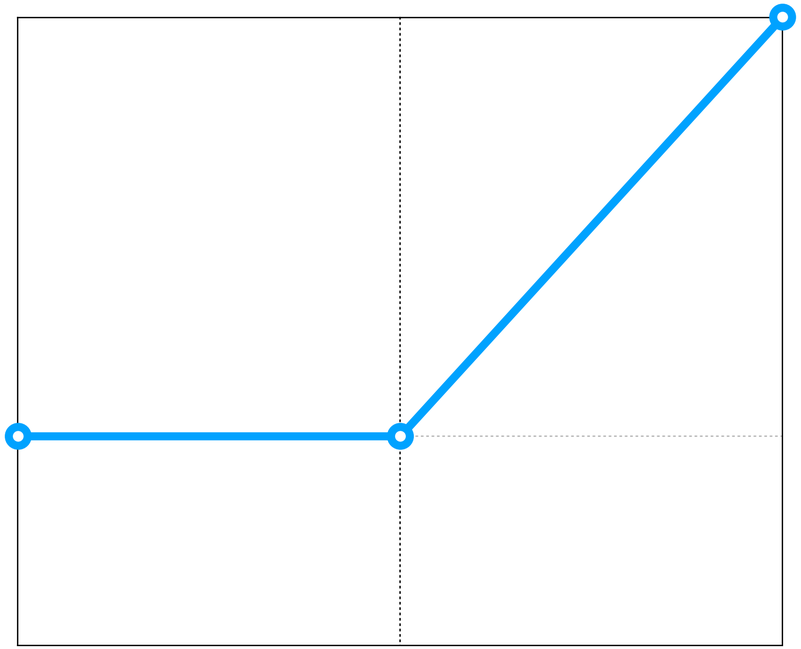
Large ML models are expensive to process

- Neural Networks with tens of layers
- Decision Tree ensembles with 100s of trees

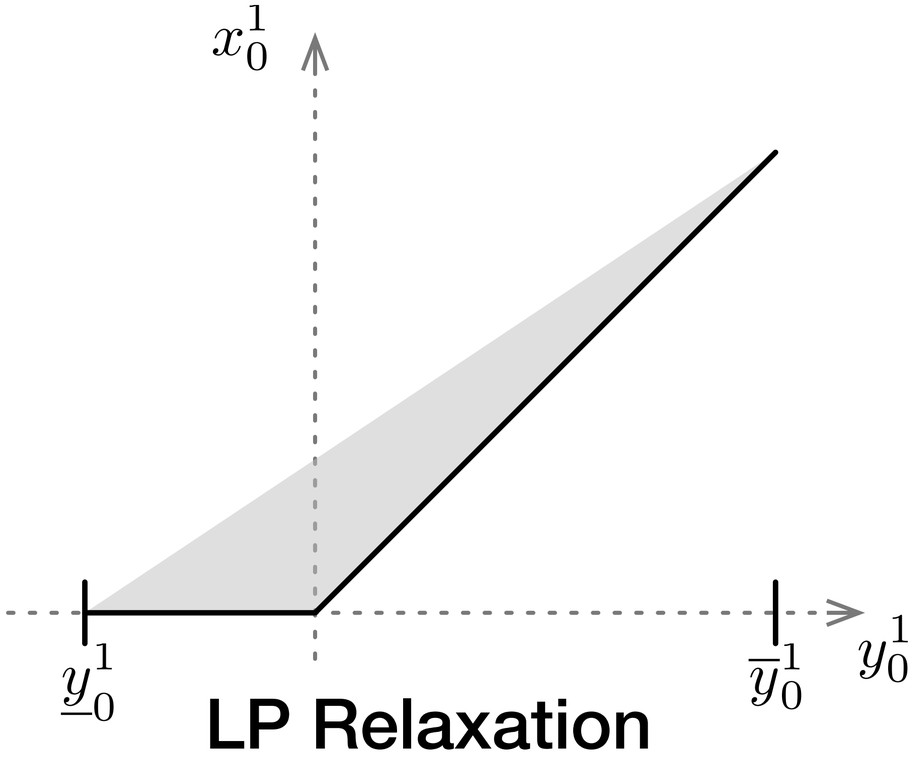
Weaker reasoning

Size Does Matter

Consider a MILP encoding for Neural Networks



True ReLU

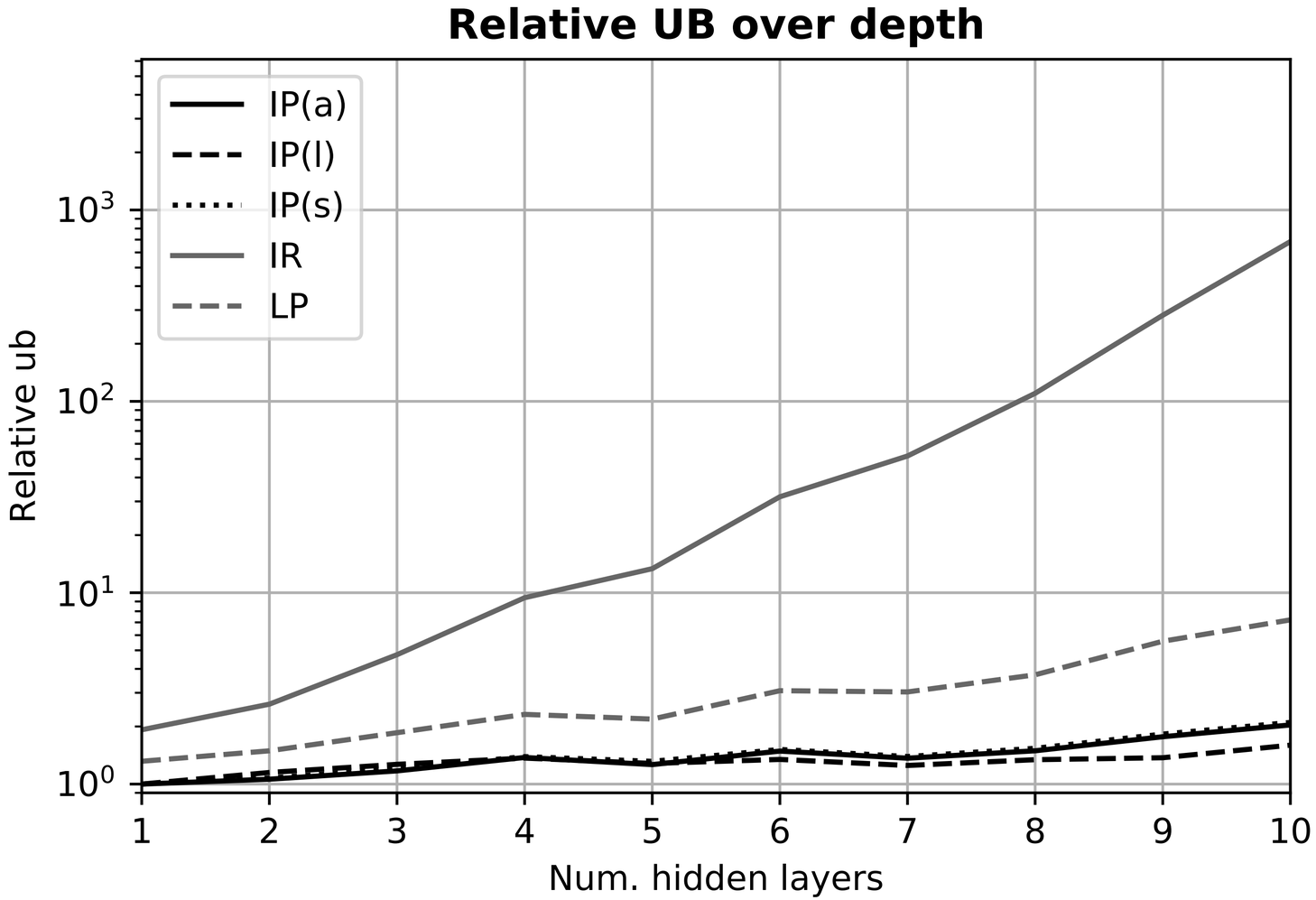


LP Relaxation

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

Size Does Matter

Some experimental data:



Bound tightening helps
(but just a bit…)

Size Does Matter

We have a scalability problem:

Large ML models are expensive to process

- Neural Networks with tens of layers
- Decision Tree ensembles with 100s of trees

Weaker reasoning

- E.g. bound degradation
- Overly complex conflicts/explanations

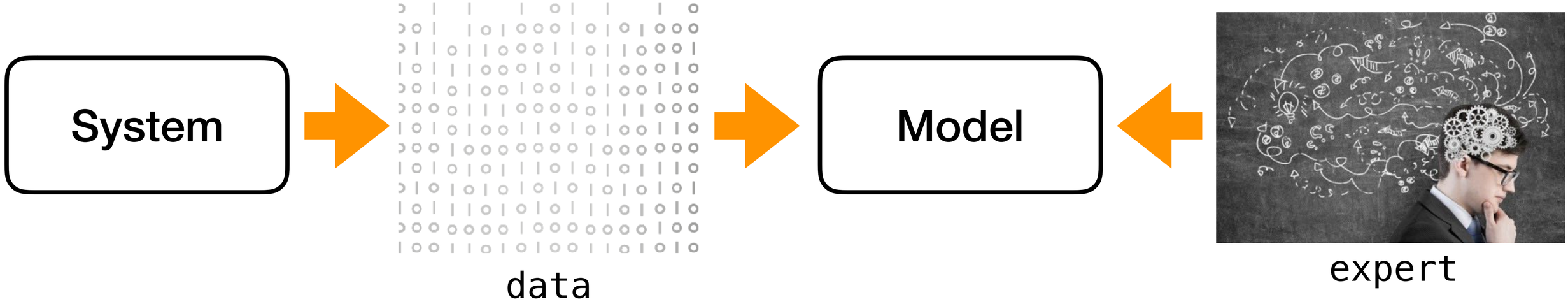
Increasing dataset size

- E.g. many inputs ➜ many examples to learn something useful

There are the usual solutions,
but also something more interesting

Subtle Implications of Approximate Models

With most real world problems, this is our situation:

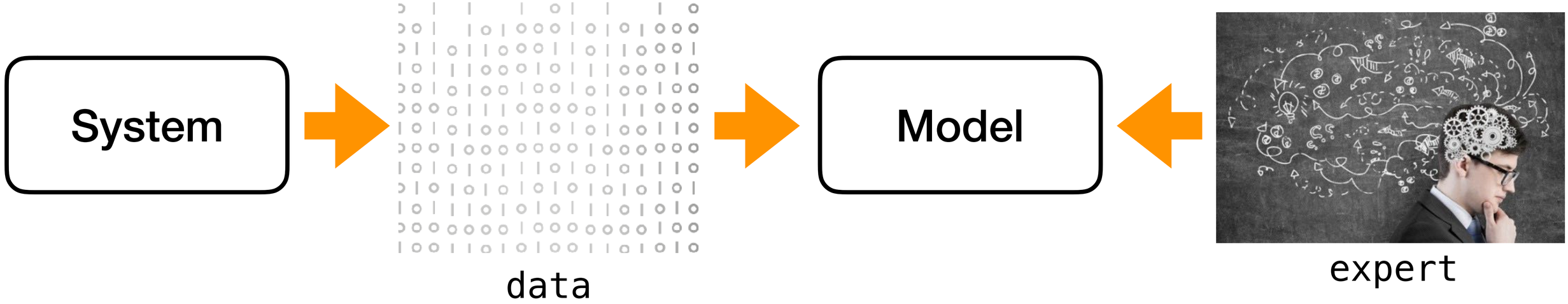


What kind of data?

- Simple parameters (e.g. costs, travel times)
- Observed values & predictions (e.g. weather & tourists)
- Decidable values & outputs (e.g. traffic lights & traffic)

Subtle Implications of Approximate Models

With most real world problems, this is our situation:



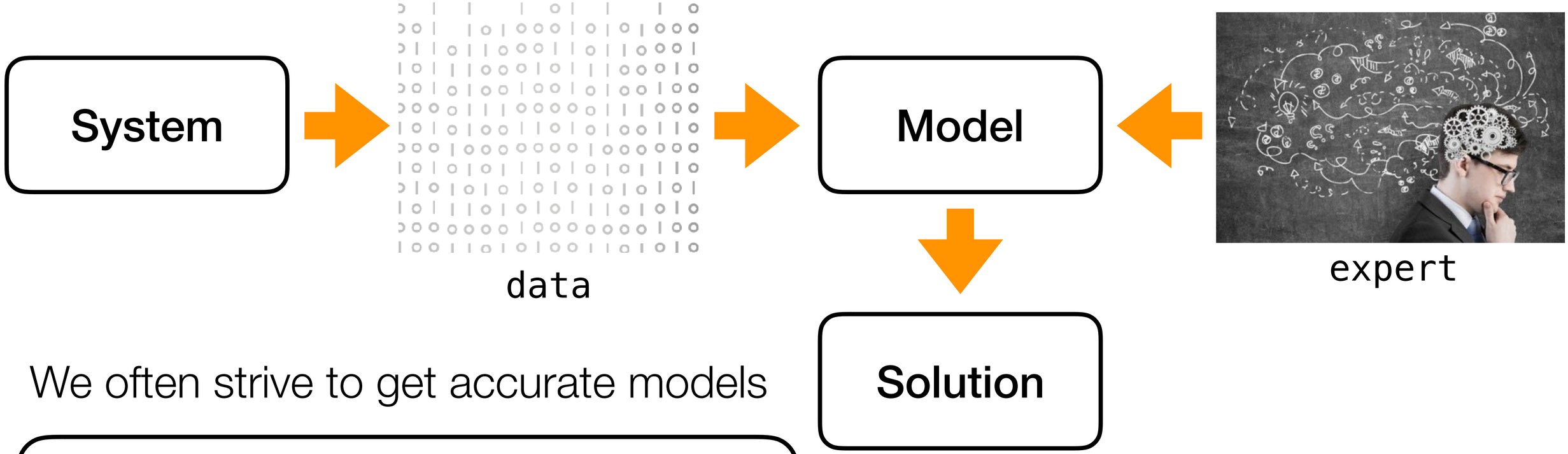
Approximation leads to a couple of problems

- Estimation errors may mislead the solver!
- The solver may find the weak spots of the ML model!

Conventional models are not immune!

Subtle Implications of Approximate Models

With most real world problems, this is our situation:

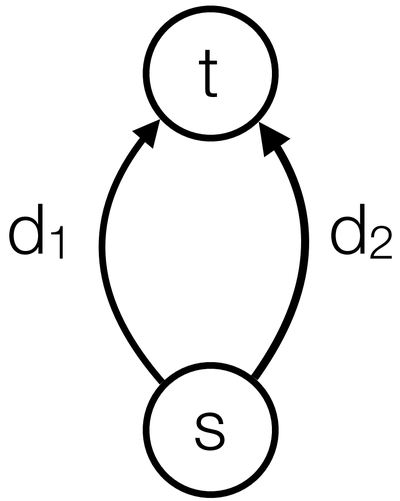


We often strive to get accurate models

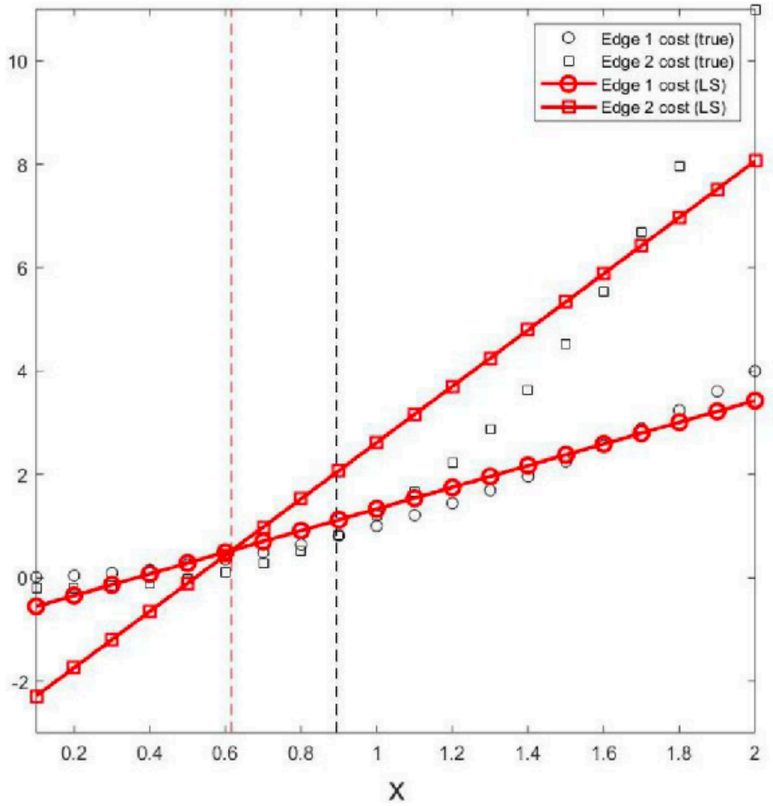
But we actually care about solutions!

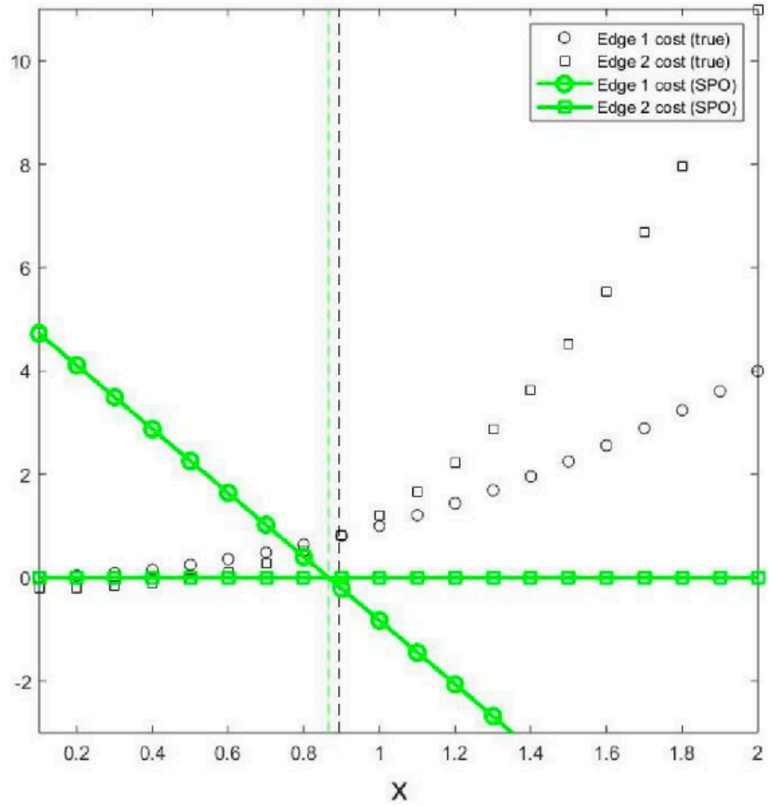
Subtle Implications of Approximate Models

An example from the literature (Smart “Predict, then Optimize”)



- x is observed
- d_1, d_2 are predicted





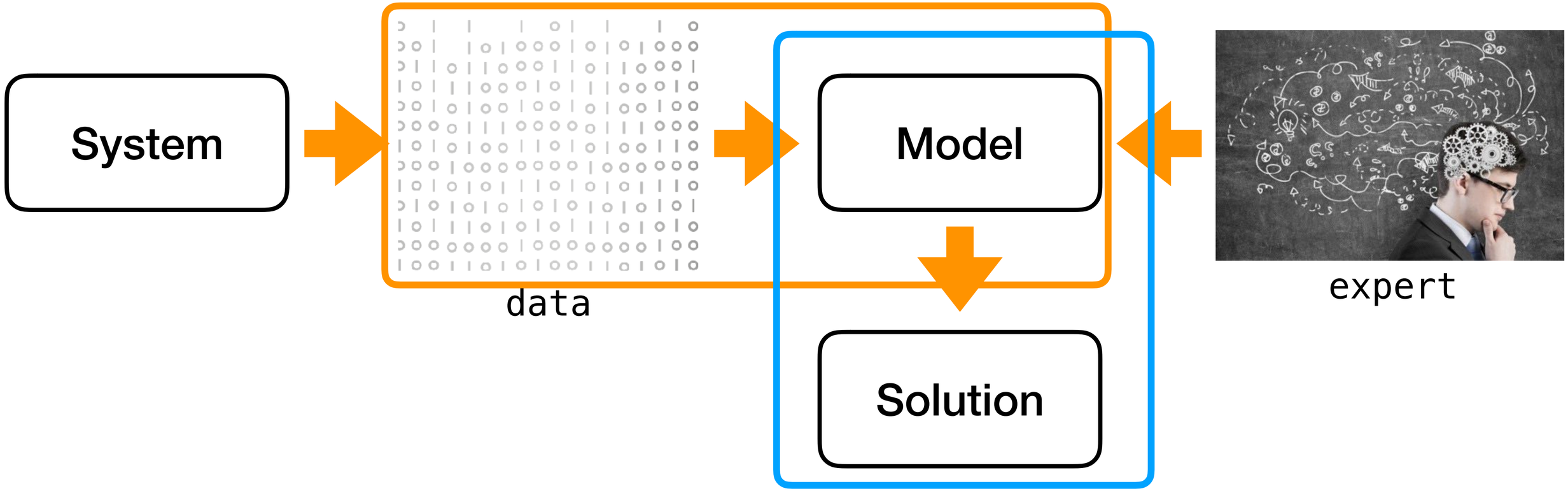
A wrong model may give a better solution

“All models are wrong,
but some are useful.”

(George Box)

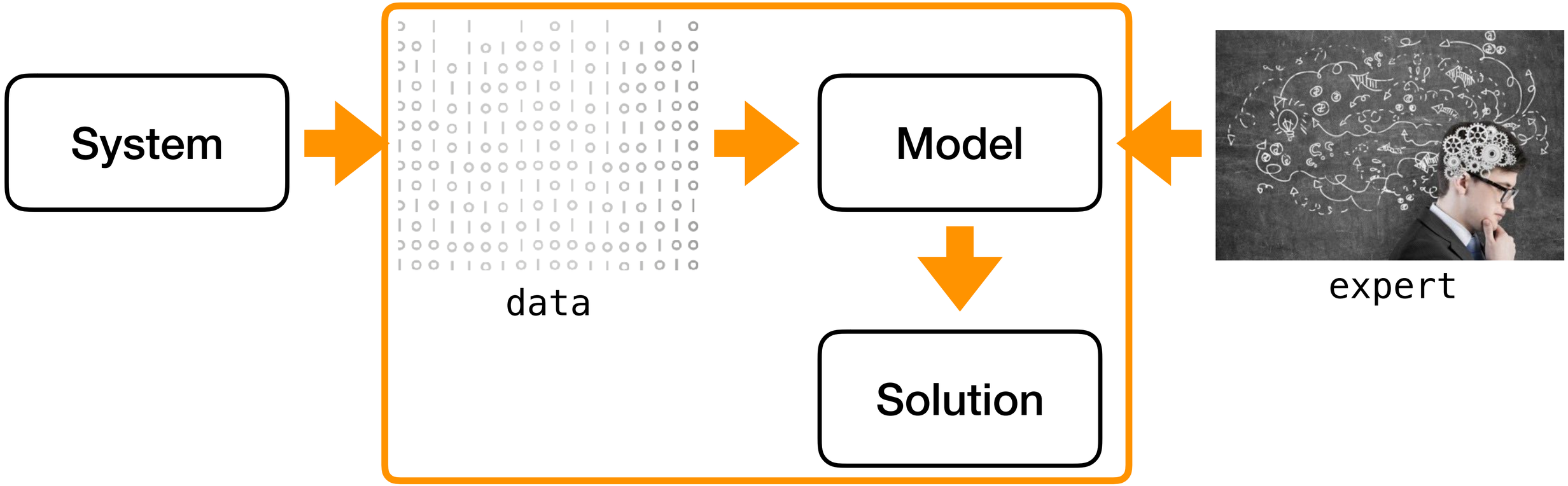
All Models Are Wrong, but Some are Useful

This is what we usually do:



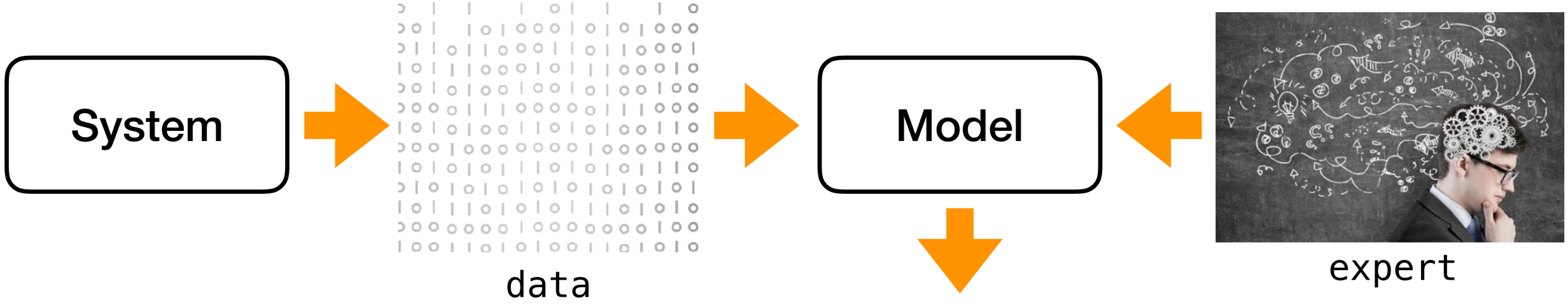
All Models Are Wrong, but Some are Useful

…But can we search for the right sort of wrong model?



All Models Are Wrong, but Some are Useful

…But can we search for the right sort of wrong model?



This “wrong" model

- Only cares about the results
- Only care about the feasible region

It could be simpler and yield better solutions!

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

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
- Smart “Predict, then Optimize”…

We made a survey (to be updated soon)!

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!)

If you like these topics…

…Do your best to attend these invited talks!

Integrating Machine Learning and Discrete Optimization
Bistra Dilkina
Thursday, Otc 3

Verification and Explanation of Deep Neural Networks
Nina Narodytska
Friday, Otc 4

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

That's all!
Got questions?

(I certainly do)

http://emlopt.github.io

www.unibo.it