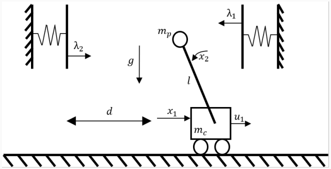
Algorithms for Identifying Flagged and Guarded Linear Systems

Guillaume Berger[1],
Monal Narasimhamurthy[2] and Sriram Sankaranarayanan[2].

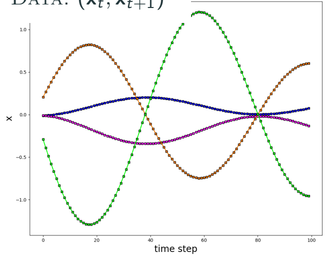
July 12, 2024

1. Université Catholique Louvain, Belgium
2. University of Colorado Boulder, USA

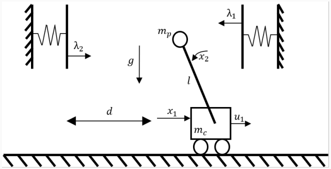
At a Glance

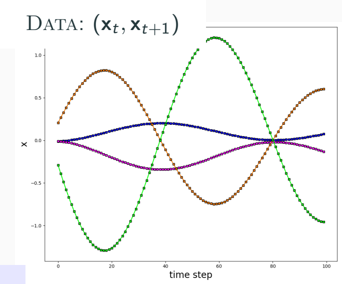


Data: $(\mathbf{x}_t, \mathbf{x}_{t+1})$



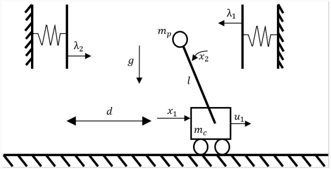
At a Glance




DATA: $(\mathbf{x}_t, \mathbf{x}_{t+1})$

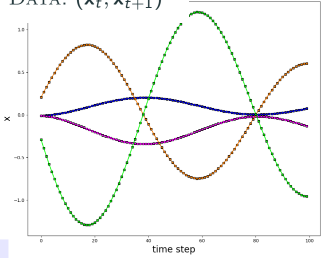
$$\begin{aligned}
x' &= x + 0.01v \\
\theta' &= \theta + 0.01\omega \\
v' &= (3.99x + 1.85\theta + 0.42v + 2.16\omega - 0.69) + \\
&[g_1]\ (-4.72x - 2.13\theta + 0.61v - 2.22\omega + 0.64) + \\
&[g_2]\ (0.49x - 0.3\theta + 0.01\omega + 0.05) \\
\omega' &= (1.82x + 0.44\theta - 0.23v + 1.9\omega - 0.31) + \\
&[g_1](-2.12x - 0.96\theta + 0.27v - \omega + 0.29) + \\
&[g_2](0.22x - 0.13\theta + 0.01\omega + 0.02)
\end{aligned}$$

At a Glance



$$
\begin{aligned}
x' &= x + 0.01v \\
\theta' &= \theta + 0.01\omega \\
v' &= (3.99x + 1.85\theta + 0.42v + 2.16\omega - 0.69) + \\
&\quad [g_1]\ (-4.72x - 2.13\theta + 0.61v - 2.22\omega + 0.64) + \\
&\quad [g_2]\ (0.49x - 0.3\theta + 0.01\omega + 0.05) \\
\omega' &= (1.82x + 0.44\theta - 0.23v + 1.9\omega - 0.31) + \\
&\quad [g_1](-2.12x - 0.96\theta + 0.27v - \omega + 0.29) + \\
&\quad [g_2](0.22x - 0.13\theta + 0.01\omega + 0.02)
\end{aligned}
$$

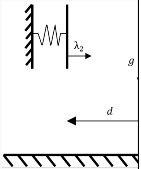
DATA: $(\mathbf{x}_t, \mathbf{x}_{t+1})$



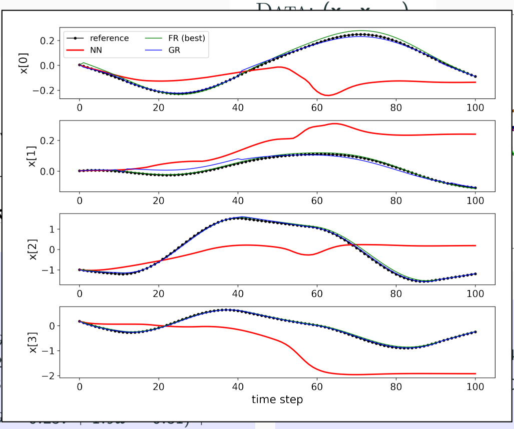
$$
\begin{aligned}
g_1(\mathbf{x}) &= -x + \theta + 0.11v - 0.48\omega + 1 \\
g_2(\mathbf{x}) &= 0.51x - \theta + 0.02v + 0.12\omega + 0.11
\end{aligned}
$$

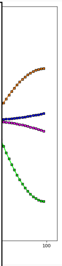
Guarded Linear System.

At a Glance



Data: $(x \quad x \quad)$





$$\begin{aligned}
x' &= x + 0.01v \\
\theta' &= \theta + 0.01\omega \\
v' &= (3.99x + 1.85\theta \\
&\quad [g_1]\ (-4.72x - 2 \\
&\quad [g_2]\ (0.49x - 0.3 \\
\omega' &= (1.82x + 0.44\theta \\
&\quad [g_1](-2.12x - 0.96\theta + 0.27v - \omega + 0.29) + \\
&\quad [g_2](0.22x - 0.13\theta + 0.01\omega + 0.02)
\end{aligned}$$

$$\begin{aligned}
&8\omega + 1 \\
&.12\omega + 0.11
\end{aligned}$$

Guarded Linear System.

Outline

- Flagged and Guarded Linear System Identification
- Expressivity
- Identification Algorithm
- Complexity Analysis
- Numerical Results.

Flagged Linear Systems

State-Variables: $(x_1, \ldots, x_n) \in \mathbb{R}^n$

Flags: $(f_1, \ldots, f_k) \in \{-1, 1\}^k$

$$\mathbf{x}(t+1) = A_0 \mathbf{x}(t) + f_1(t) \times A_1 \mathbf{x}(t) + \cdots + f_k(t) \times A_m \mathbf{x}(t)\,.$$

Flagged Linear Systems

State-Variables: $(x_1, \ldots, x_n) \in \mathbb{R}^n$

Flags: $(f_1, \ldots, f_k) \in \{-1, 1\}^k$

$$\mathbf{x}(t+1) = A_0\mathbf{x}(t) + f_1(t) \times A_1\mathbf{x}(t) + \cdots + f_k(t) \times A_m\mathbf{x}(t)\,.$$

Linear Switched System:

- Exogenous Switching Signal (flags $f_i(t) \in \{-1, 1\}$).
- 2^k modes.

Guarded Linear Systems

State-Variables: $(x_1, \ldots, x_n) \in \mathbb{R}^n$

$$\mathbf{x}(t+1) = A_0\mathbf{x}(t) + [g_1(\mathbf{x}(t))] \times A_1\mathbf{x}(t) + \cdots + [g_m(\mathbf{x}(t))] \times A_m\mathbf{x}(t)\,.$$

Guarded Linear Systems

State-Variables: $(x_1, \ldots, x_n) \in \mathbb{R}^n$

$$\mathbf{x}(t+1) = A_0\mathbf{x}(t) + [g_1(\mathbf{x}(t))] \times A_1\mathbf{x}(t) + \cdots + [g_m(\mathbf{x}(t))] \times A_m\mathbf{x}(t)\,.$$

Guards:

- $g_i(\mathbf{x}) = \mathbf{c}_i^\top \mathbf{x} + d_i$
- $[g_i(\mathbf{x})] = \begin{cases} +1 & g_i(\mathbf{x}) \geq 0 \\ -1 & o.w. \end{cases}$

Guarded Linear Systems

State-Variables: $(x_1, \ldots, x_n) \in \mathbb{R}^n$

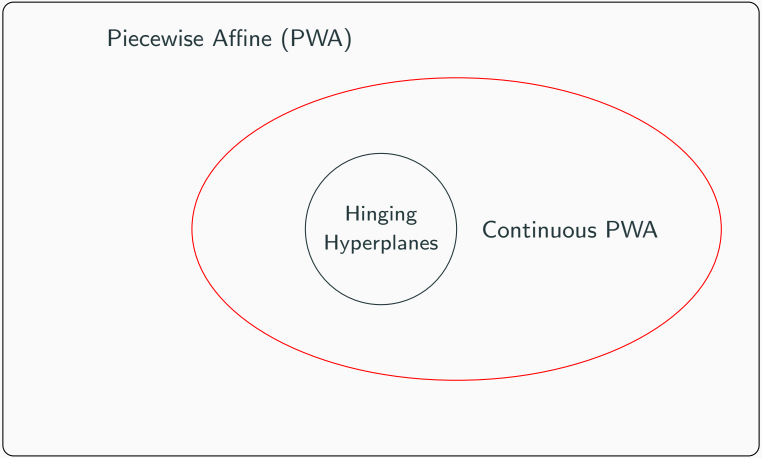
$$\mathbf{x}(t+1) = A_0\mathbf{x}(t) + [g_1(\mathbf{x}(t))] \times A_1\mathbf{x}(t) + \cdots + [g_m(\mathbf{x}(t))] \times A_m\mathbf{x}(t)\,.$$

Guards:

- $g_i(\mathbf{x}) = \mathbf{c}_i^\top \mathbf{x} + d_i$
- $[g_i(\mathbf{x})] = \begin{cases} +1 & g_i(\mathbf{x}) \geq 0 \\ -1 & o.w. \end{cases}$

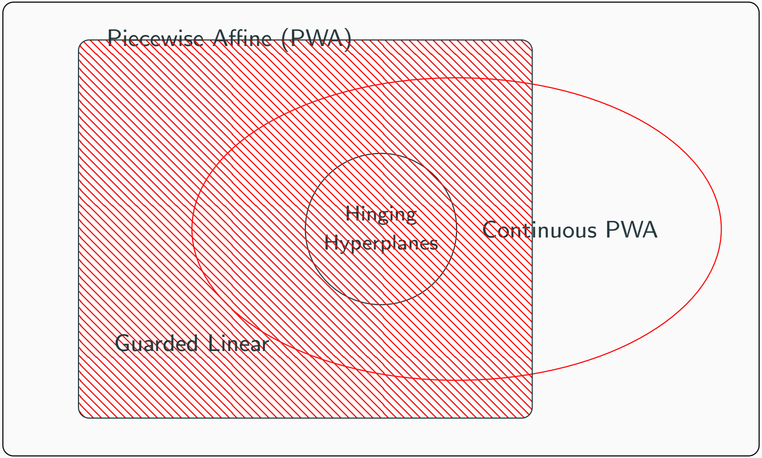
Compactly specified state-based switching.

Expressivity



Hinging Hyperplanes [Brieman'93]:
$f(\mathbf{x}) = c_0^t\mathbf{x} + \sum_{j=1}^{k} \pm \max(\mathbf{c}_j^t\mathbf{x}, \mathbf{d}_j^t\mathbf{x})$

Expressivity



Hinging Hyperplanes [Brieman'93]:
$f(\mathbf{x}) = c_0^t\mathbf{x} + \sum_{j=1}^{k} \pm \max(\mathbf{c}_j^t\mathbf{x}, \mathbf{d}_j^t\mathbf{x})$

Flagged Linear Regression

Input:

1. Data $(\mathbf{x}_t, \mathbf{y}_t)$, $t = 1, \ldots, N$
2. $\#$ of flags: k
3. Relative Error : $\epsilon > 0$, Absolute Error: $\tau > 0$

Flagged Linear Regression

Input:

1. Data $(\mathbf{x}_t, \mathbf{y}_t)$, $t = 1, \ldots, N$
2. $\#$ of flags: k
3. Relative Error : $\epsilon > 0$, Absolute Error: $\tau > 0$

Output:

1. Matrices $A_0, \ldots, A_k$
2. Latent Flags: $f_t^{(1)}, \ldots, f_t^{(k)}$.

Flagged Linear Regression

Input:

1. Data $(\mathbf{x}_t, \mathbf{y}_t)$, $t = 1, \ldots, N$
2. $\#$ of flags: k
3. Relative Error : $\epsilon > 0$, Absolute Error: $\tau > 0$

Output:

1. Matrices $A_0, \ldots, A_k$
2. Latent Flags: $f_t^{(1)}, \ldots, f_t^{(k)}$.

$$(\forall t)\ ||\mathbf{y}_t - (A_0\mathbf{x}_t + \textstyle\sum_{j=1}^{k} f_t^{(j)} A_j \mathbf{x}_t)|| \leq \epsilon||\mathbf{x}_t|| + \tau$$

Flagged Linear Regression

Input:

1. Data $(\mathbf{x}_t, \mathbf{y}_t)$, $t = 1, \ldots, N$
2. $\#$ of flags: k
3. Relative Error : $\epsilon > 0$, Absolute Error: $\tau > 0$

Output:

1. Matrices $A_0, \ldots, A_k$
2. Latent Flags: $f_t^{(1)}, \ldots, f_t^{(k)}$.

$$(\forall t)\ \underbrace{||\mathbf{y}_t - (A_0\mathbf{x}_t + \textstyle\sum_{j=1}^{k} f_t^{(j)} A_j \mathbf{x}_t)||}_{\text{Prediction error: } (\mathbf{x}_t, \mathbf{y}_t)} \leq \epsilon ||\mathbf{x}_t|| + \tau$$

Flagged Linear Regression

Input:

1. Data $(\mathbf{x}_t, \mathbf{y}_t)$, $t = 1, \ldots, N$
2. $\#$ of flags: k
3. Relative Error : $\epsilon > 0$, Absolute Error: $\tau > 0$

Output:

1. Matrices $A_0, \ldots, A_k$
2. Latent Flags: $f_t^{(1)}, \ldots, f_t^{(k)}$.

$$(\forall t)\ \underbrace{||\mathbf{y}_t - (A_0\mathbf{x}_t + \sum_{j=1}^{k} f_t^{(j)} A_j\mathbf{x}_t)||}_{\text{Prediction error: } (\mathbf{x}_t, \mathbf{y}_t)} \leq \overset{\text{Rel. Err.}}{\boxed{\epsilon||\mathbf{x}_t||}} + \tau$$

Approximation Algorithm

Goal: Find model that minimizes rel. error ϵ.

Fix: abs. error τ and # latent flags k.

Best Known Algorithm: Mixed Integer Linear Programming.

Complexity: $O(\boxed{2^{kN}} \times \text{poly}(N, k, n))$.

Approximation Algorithm:

Guaranteed solution in the interval $[\epsilon^*, \epsilon^* + \epsilon_{\text{gap}}]$.

Main Contribution

Inputs: Data $(\mathbf{x}_t, \mathbf{y}_t)$ for $t \in [N]$, $\#$ Flags k, Err. gap $\epsilon_{\mathsf{gap}}, \tau > 0$.

Approximation Algorithm:

- Guaranteed solution in the interval $[\epsilon^*, \epsilon^* + \epsilon_{\mathsf{gap}}]$.

Main Contribution

Inputs: Data $(\mathbf{x}_t, \mathbf{y}_t)$ for $t \in [N]$, $\#$ Flags k, Err. gap $\epsilon_{\text{gap}}, \tau > 0$.

Approximation Algorithm:

- Guaranteed solution in the interval $[\epsilon^*, \epsilon^* + \epsilon_{\text{gap}}]$.
- Running time complexity:

$$O\left(2^{O\left(k^3 n^4 \log\left(\frac{k}{\epsilon_{\text{gap}}}\right)\right)} \mathrm{poly}\left(k, n, \log\left(\frac{1}{\epsilon_{\text{gap}}}\right)\right) \times N\right)$$

Main Contribution

Inputs: Data $(\mathbf{x}_t, \mathbf{y}_t)$ for $t \in [N]$, $\#$ Flags k, Err. gap $\epsilon_{\text{gap}}, \tau > 0$.

Approximation Algorithm:

- Guaranteed solution in the interval $[\epsilon^*, \epsilon^* + \epsilon_{\text{gap}}]$.
- Running time complexity:

$$O\left(2^{O\left(k^3 n^4 \log\left(\frac{k}{\epsilon_{\text{gap}}}\right)\right)} \text{poly}\left(k, n, \log\left(\frac{1}{\epsilon_{\text{gap}}}\right)\right) \times N\right)$$

- Linear in the size of the data.

Main Contribution

Inputs: Data $(\mathbf{x}_t, \mathbf{y}_t)$ for $t \in [N]$, # Flags k, Err. gap $\epsilon_{\text{gap}}, \tau > 0$.

Approximation Algorithm:

- Guaranteed solution in the interval $[\epsilon^*, \epsilon^* + \epsilon_{\text{gap}}]$.
- Running time complexity:

$$O\left(2^{O\left(k^3 n^4 \log\left(\frac{k}{\epsilon_{\text{gap}}}\right)\right)} \mathrm{poly}\left(k, n, \log\left(\frac{1}{\epsilon_{\text{gap}}}\right)\right) \times N\right)$$

- Linear in the size of the data.
- Compared to [Berger et al. Neurips 2022]: $O(2^{k^3})$ vs. $O(2^{k \times 2^k})$.

Main Contribution

Inputs: Data $(\mathbf{x}_t, \mathbf{y}_t)$ for $t \in [N]$, # Flags k, Err. gap $\epsilon_{\text{gap}}, \tau > 0$.

Approximation Algorithm:

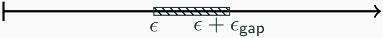
- Guaranteed solution in the interval $[\epsilon^*, \epsilon^* + \epsilon_{\text{gap}}]$.
- Running time complexity:

$$O\left(2^{O\left(k^3 n^4 \log\left(\frac{k}{\epsilon_{\text{gap}}}\right)\right)} \mathrm{poly}\left(k, n, \log\left(\frac{1}{\epsilon_{\text{gap}}}\right)\right) \times N\right)$$

- Linear in the size of the data.
- Compared to [Berger et al. Neurips 2022]: $O(2^{k^3})$ vs. $O(2^{k \times 2^k})$.
- "Simple" algorithm inspired by CEGIS and Branch-and-Cut.

Promise Problem

Decision (Yes/No) version of an approximation algorithm [Even+Selman+Yacobi'82, Goldreich'2006].

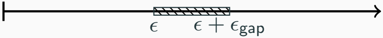


Assume:

- ϵ^* is optimal relative err.
- $\epsilon^* \leq \epsilon$ or $\epsilon^* > \epsilon + \epsilon_{\text{gap}}$.

Promise Problem

Decision (Yes/No) version of an approximation algorithm [Even+Selman+Yacobi'82, Goldreich'2006].



Assume:

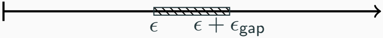
- ϵ^* is optimal relative err.
- $\epsilon^* \leq \epsilon$ or $\epsilon^* > \epsilon + \epsilon_{\text{gap}}$.

Output:

- $\textsc{Yes}$: $\epsilon^* \leq \epsilon$.
- $\textsc{No}$: $\epsilon^* > \epsilon + \epsilon_{\text{gap}}$.

Promise Problem

Decision (Yes/No) version of an approximation algorithm [Even+Selman+Yacobi'82, Goldreich'2006].



Assume:

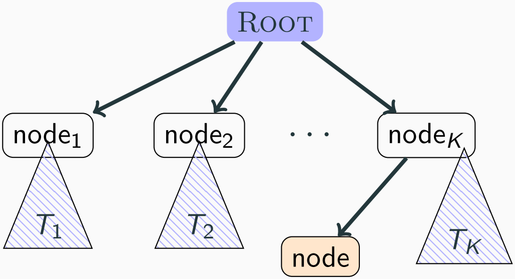
- ϵ^* is optimal relative err.
- $\epsilon^* \leq \epsilon$ or $\epsilon^* > \epsilon + \epsilon_{\text{gap}}$.

Output:

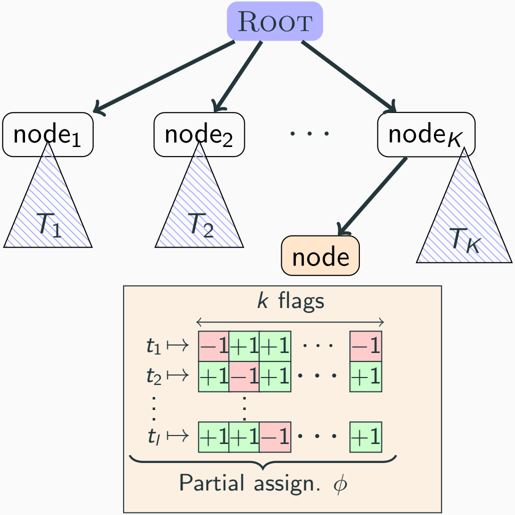
- $\textsc{Yes}$: $\epsilon^* \leq \epsilon$. $\boxed{\textit{Output model with rel. err} \leq \epsilon + \epsilon_{\textit{gap}}}$
- $\textsc{No}$: $\epsilon^* > \epsilon + \epsilon_{\text{gap}}$.

Algorithm (Flagged Regression)

Tree Search

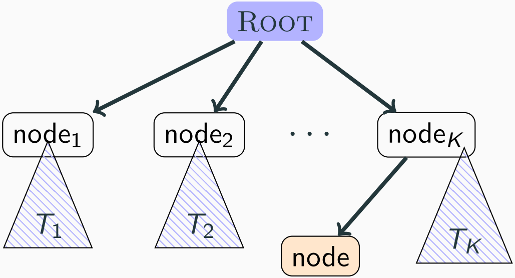


Tree Search

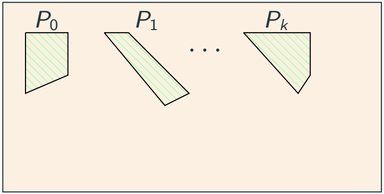


$t_1 \mapsto$	-1	$+1$	$+1$	$\cdots$	-1
$t_2 \mapsto$	$+1$	-1	$+1$	$\cdots$	$+1$
$\vdots$		$\vdots$			
$t_l \mapsto$	$+1$	$+1$	-1	$\cdots$	$+1$

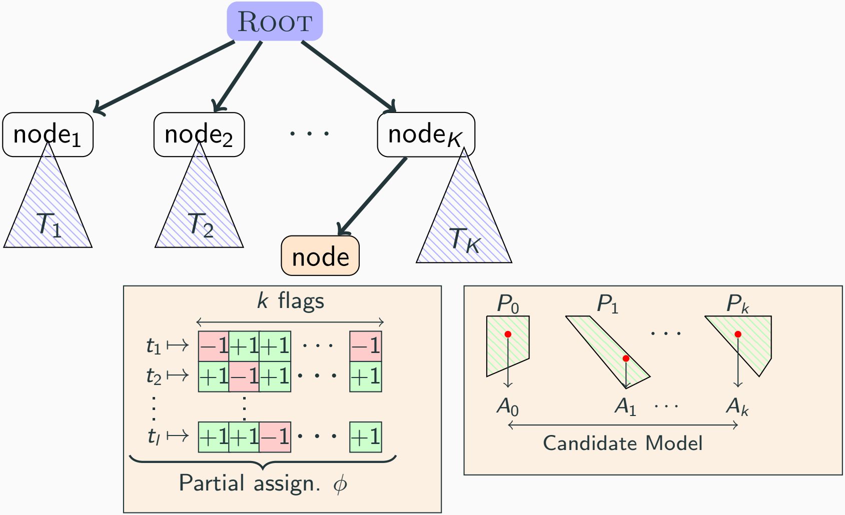
Tree Search



$$\underbrace{\begin{array}{rcccccc} & & \multicolumn{5}{c}{\overleftrightarrow{k\ \text{flags}}} \\ t_1 & \mapsto & -1 & +1 & +1 & \cdots & -1 \\ t_2 & \mapsto & +1 & -1 & +1 & \cdots & +1 \\ \vdots & & & \vdots & & & \\ t_l & \mapsto & +1 & +1 & -1 & \cdots & +1 \end{array}}_{\text{Partial assign. } \phi}$$

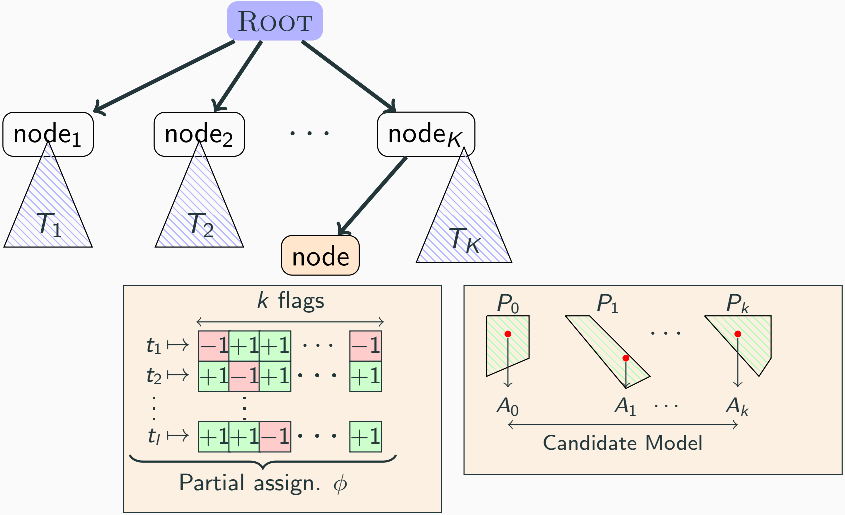


Tree Search



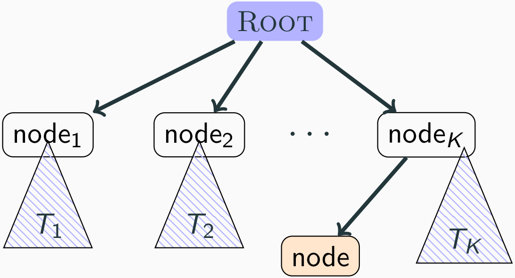
$t_1 \mapsto$	-1	$+1$	$+1$	$\cdots$	-1
$t_2 \mapsto$	$+1$	-1	$+1$	$\cdots$	$+1$
$\vdots$		$\vdots$			
$t_l \mapsto$	$+1$	$+1$	-1	$\cdots$	$+1$

Tree Search Algorithm: Expanding a Node



$t_1 \mapsto$	-1	$+1$	$+1$	$\cdots$	-1
$t_2 \mapsto$	$+1$	-1	$+1$	$\cdots$	$+1$
$\vdots$		$\vdots$			
$t_l \mapsto$	$+1$	$+1$	-1	$\cdots$	$+1$

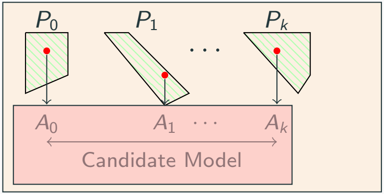
Tree Search Algorithm: Expanding a Node



k flags

$t_1 \mapsto$	-1	$+1$	$+1$	$\cdots$	-1
$t_2 \mapsto$	$+1$	-1	$+1$	$\cdots$	$+1$
$\vdots$		$\vdots$			
$t_l \mapsto$	$+1$	$+1$	-1	$\cdots$	$+1$

Partial assign. ϕ

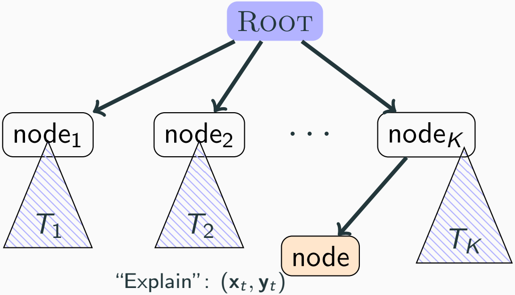


Explains remaining points?

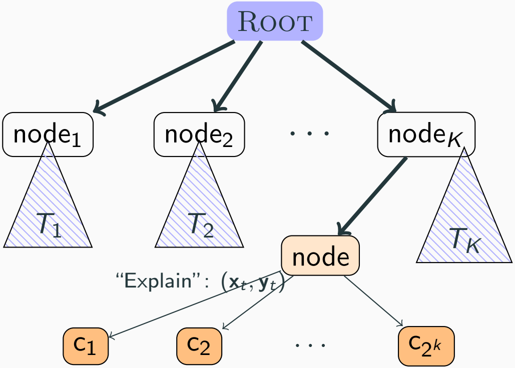
Yes!

No: $(\mathbf{x}_t, \mathbf{y}_t)$

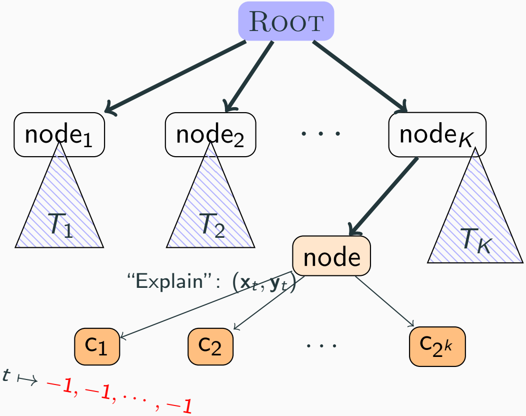
Expand Leaf Node # 2



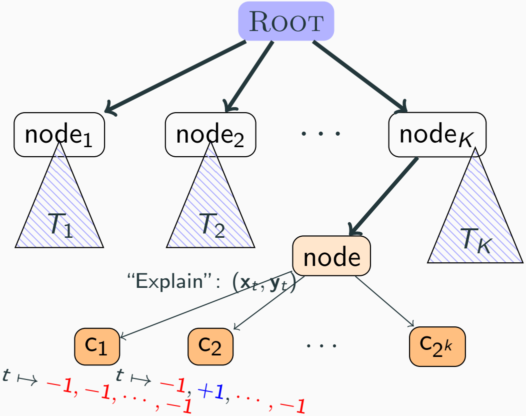
Expand Leaf Node # 2



Expand Leaf Node # 2



Expand Leaf Node # 2

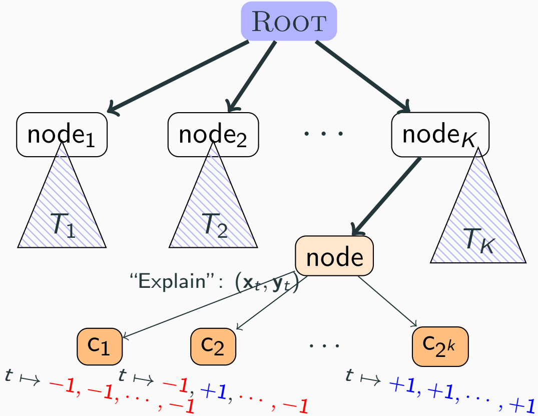


“Explain”: $(\mathbf{x}_t, \mathbf{y}_t)$

$t \mapsto -1, -1, \ldots, -1$

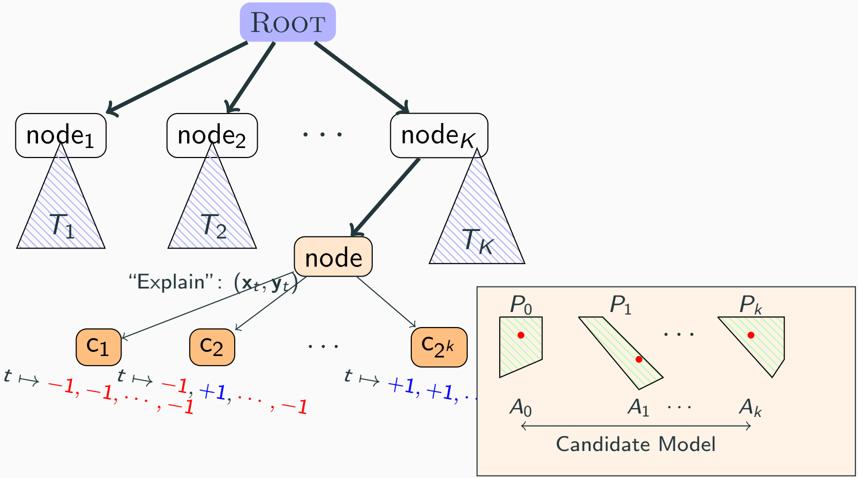
$t \mapsto -1, +1, \ldots, -1$

Expand Leaf Node # 2

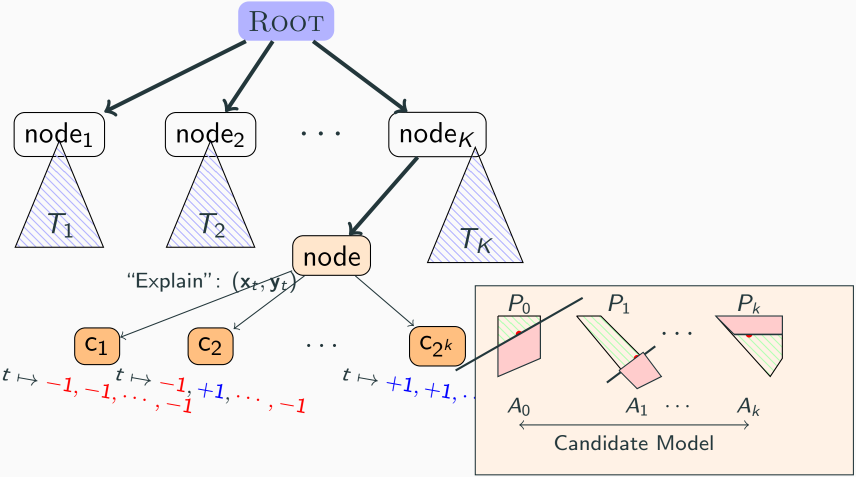


Root
node$_1$
node$_2$
$\cdots$
node$_K$
T_1
T_2
T_K
node
"Explain": $(\mathbf{x}_t, \mathbf{y}_t)$
c$_1$
c$_2$
$\cdots$
c$_{2^k}$
$t \mapsto -1, -1, \dots, -1$
$t \mapsto -1, +1, \dots, -1$
$t \mapsto +1, +1, \dots, +1$

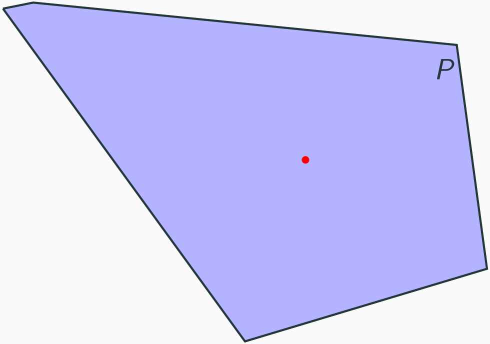
Expand Leaf Node # 2



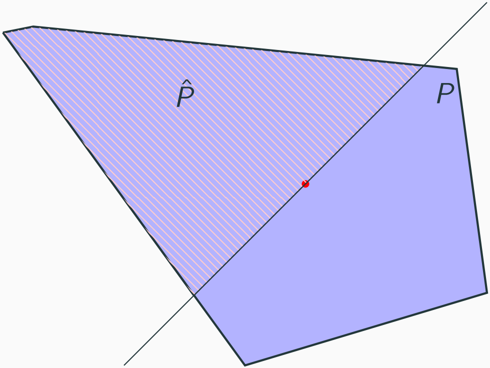
Expand Leaf Node # 2



Cutting Plane Method

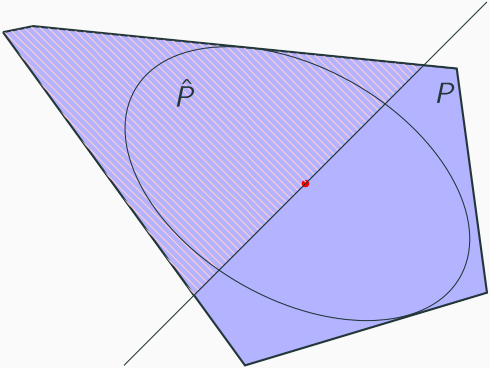


Cutting Plane Method



How does vol($\hat{P}$) relate to vol(P)?

Cutting Plane Method



Center of Max. Vol. Inscribed Ellipsoid:

$$\mathrm{vol}(\hat{P}) \leq \left(1 - \frac{1}{2n}\right) \mathrm{vol}(P).$$

Overall Algorithm

- Expand Tree by choosing unexplained data point.
- Choose MVE center $\Rightarrow$ volume of one polyhedron shrinks.
- *Key Result:* If $\epsilon^* \leq \epsilon$ then $\exists$ node with $P \supseteq \mathcal{B}(L(\epsilon_{\text{gap}}))$.
    - If *Chebyshev Radius* $< R(\epsilon_{\text{gap}})$ node can be pruned.

Overall Algorithm

- Expand Tree by choosing unexplained data point.
- Choose MVE center ⇒ volume of one polyhedron shrinks.
- *Key Result:* If $\epsilon^* \leq \epsilon$ then $\exists$ node with $P \supseteq \mathcal{B}(L(\epsilon_{\text{gap}}))$.
  - If *Chebyshev Radius* $< R(\epsilon_{\text{gap}})$ node can be pruned.

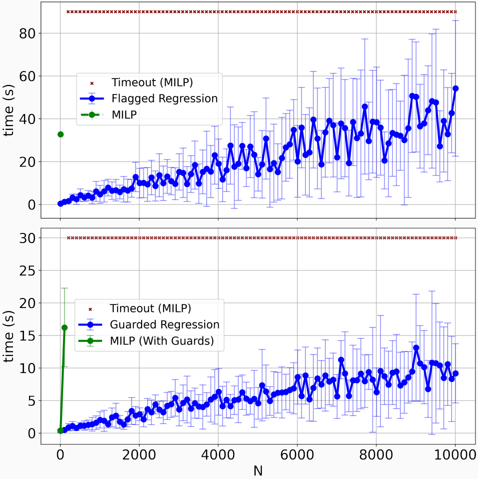
- *Guarantee:* If $\epsilon^* \leq \epsilon$ then guaranteed to find a model with error $\leq \epsilon + \epsilon_{\text{gap}}$.
- Depth of the tree is bounded.
- Overall Time Complexity:

$$2^{O\left(k^3 n^4 \log\left(\frac{k}{\epsilon_{\text{gap}}}\right)\right)} \mathrm{poly}\left(k, n, \log\left(\frac{1}{\epsilon_{\text{gap}}}\right)\right) N$$

Empirical Evaluation

Microbenchmarks

Randomly generated models: $n = 2, k = 3$ with $N \in [1, 10^4]$.

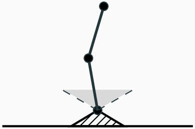


Experimental Comparisons

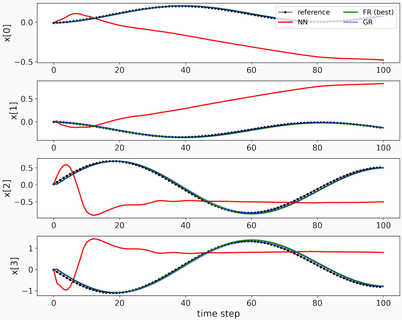
- Benchmarks: acrobot, cartpole with soft-walls and 6-DOF industrial robot arm.
- Comparison with Neural Networks:
    - Feedforward models: 2 layes, 32 RELU units/layer.
    - Trained using Tensorflow.
    - Training error $< 10^{-4}$.
- Comparison with PARC: [Bemporad 2022].
    - We set number of regions $K = 10$.
    - Other hyper-parameters were set as recommended by the manual.

Acrobot with Soft Joints

- Multiple arms connected by joints and “soft” walls [Aydinoglu et al. 2021].
- Generated data from simulations and added random noise to states.
- Comparisons: guarded linear regression versus neural network learning.



Acrobot Comparisons

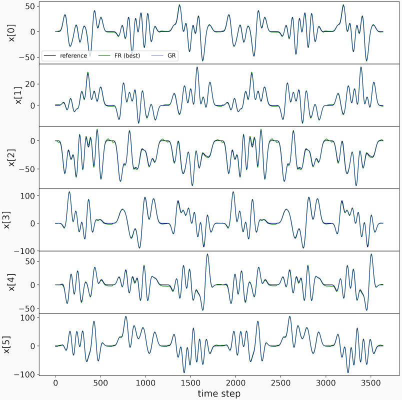


	N=200		N=400		N=800		N=1000	
	R^2	t(s)	R^2	t(s)	R^2	t(s)	R^2	t(s)
NN	-0.75	1.90	0.74	2.84	0.87	5.17	0.89	6.9
PARC	-0.95	1.34	0.94	4.23	0.99	6.9	0.96	7.04
FR	0.99	2.25	0.99	7.6	0.99	8.41	0.99	11.5
GR	0.99	13.16	0.99	12.0	0.92	21.8	0.99	19.1

6-DOF Robot Arm

- States collected from a 6-DOF industrial robot arm [Weigand et al. 2023].
- Nonlinear System Identification benchmark.
- 6 state variables and 6 control inputs.
- Flagged/Guarded regression $k = 4$.

6-DOF Robot Arm



Approach	Test NMRSE	R^2 score	Time (s)
Linear	0.83	0.31	unspecified
NN	0.30	0.88	3.02
PARC	1.78	-7.63	27.71
FR	0.14	0.98	82.32

Conclusion and Future Work

- Flagged/Guarded Linear Systems may be useful.
- Approximate identification algorithm with guarantees.
  - Guaranteed approximation error for relative error tolerance.
  - Linear in number of data points.
  - Exponential in number of flags/dimensions.
- Implementation runs in few minutes for $n \leq 12$, $k \leq 4$.
- *How do we take it to the next level*?
- *Future Work*.
  - Incorporate more system knowledge/constraints to speed up identification?
  - PCA-style estimation [Vidal+Ma+Sastry'2005].

Thank You!

G. Berger was supported in part by a fellowship from the Belgian American Educational Foundation (BAEF). This work was funded by the US National Science Foundation (NSF) under grant numbers CPS 1836900 and 1932189.