

TRAINING RECURRENT NEURAL-NETWORK MODELS ON THE INDUSTRIAL ROBOT DATASET UNDER ℓ_1 -REGULARIZATION

Alberto Bemporad

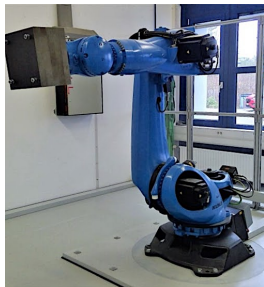
imt.lu/ab

- Industrial robot benchmark problem: description and challenges
- Recurrent neural network model in residual form
- Training methods supporting ℓ_1 -penalties for sparsification
- Industrial robot benchmark problem: identification results
- Conclusions

INDUSTRIAL ROBOT BENCHMARK

(Weigand, Götz, Ulmen, Ruskowski, 2022)

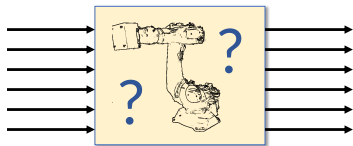
- KUKA KR300 R2500 ultra SE industrial robot, full robot movement
- **6 inputs** (torques), **6 outputs** (joint angles), backlash
- Identification benchmark dataset (forward model):
 - Sample time: $T_s = 100$ ms
 - $N = 39988$ training samples
 - $N_{\text{test}} = 3636$ test samples



nonlinearbenchmark.org


INDUSTRIAL ROBOT BENCHMARK: CHALLENGES

- Highly **nonlinear** dynamics.
Nonlinear modeling required
- **Multi-input / multi-output**, highly coupled system
- Data are **slightly over-sampled**, $\|y_k - y_{k-1}\|$ is often very small,
need to minimize open-loop simulation error
- **Limited information**: easy to overfit training data and get poor testing results
- **Large number of samples** complicates numerical optimization



Finding a model that minimizes the simulation error is a rather challenging task from a computational viewpoint

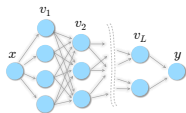
CONTROL-ORIENTED MODELS

- **Target:** get a good dynamical model for **model-based (predictive) control**:
 - Complex model = complex controller  model must be **simple**
 - Easy to **linearize** (to get Jacobian matrices for nonlinear optimization)
- Similar requirements for **Kalman filtering** (simple model, easy to linearize)
- **Metrics:** best fit on predicting test data not seen during training
- Compact **recurrent neural network** models are good candidates

RECURRENT NEURAL NETWORKS IN RESIDUAL FORM

- **Recurrent Neural Network (RNN)** model in **residual form**:

$$\begin{aligned}x_{k+1} &= Ax_k + Bu_k + f_x(x_k, u_k, \theta_x^i) \\ y_k &= Cx_k + f_y(x_k, \theta_y^i) \\ f_x, f_y &= \text{feedforward neural network}\end{aligned}$$



$$v_j = A_j f_{j-1}(v_{j-1}) + b_j$$

$$\theta = (A_1, b_1, \dots, A_L, b_L)$$

- Training problem: minimize **open-loop simulation error** under regularization

$$\begin{aligned}\min_{A, B, C, \theta_x, \theta_y} \frac{1}{N} \sum_{k=1}^N \|y_k - \hat{y}_k\|_2^2 + \frac{1}{2} \rho(\|\theta_x\|_2^2 + \|\theta_y\|_2^2) + \tau(\|\theta_x\|_1 + \|\theta_y\|_1) \\ \text{s.t. model equations, } x_0 = 0\end{aligned}$$

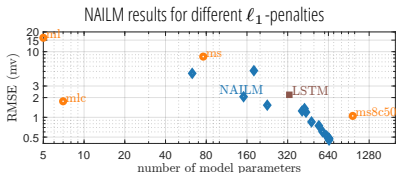
- **ℓ_1 -regularization** introduced to reduce # model coefficients (=simpler model)

TRAINING RNN W/ ℓ_1 -PENALTIES

- The **RNN training problem** w/ ℓ_1 -regularization can be solved in several ways:
 - gradient descent methods**, like Adam (Kingma, Ba, 2014) and similar variants
 - non-smooth NLP solvers**, like OWL-QN (Andrew, Gao, 2007) (general purpose)
 - extended Kalman filtering** (Bemporad, 2023) (specific to train RNNs)
 - Generalized Gauss-Newton + ADMM method (**NAILM**) (Bemporad, 2021, arXiv) ¹:

Silverbox benchmark	RMSE [mV]	BFR [%]
ARX (ml) [1]	16.29 [4.40]	69.22 [73.79]
NLARX (ms) [1]	8.42 [4.20]	83.67 [92.06]
NLARX (mlc) [1]	1.75 [1.70]	96.67 [96.79]
NLARX (ms8c50) [1]	1.05 [0.30]	98.01 [99.43]
Recurrent LSTM model [2]	2.20	95.83
SS encoder [3] ($n_x = 4$)	[1.40]	[97.35]
NAILM	0.35	99.33

≈86000 training data, 40000 test data



¹Trained RNN model available at <http://cse.lab.imtlucca.it/~bemporad/shared/silverbox/rnn888.zip>

[1] Ljung, Zhang, Lindskog, Juditski, 2004 [2] Ljung, Andersson, Tiels, Schön, 2020 [3] Beintema, Toth, Schoukens, 2021

TRAINING RNN W/ ℓ_1 -PENALTIES - INDUSTRIAL ROBOT

- Main **issues** with industrial robot benchmark:
 - **many parameters** to train, **large dataset** \Rightarrow complex NLP
 - **high sensitivity** wrt weights (dynamics gets easily unstable)
 - **local minima** (solution depends on initial guess)
 - cannot easily use **mini-batches**: open-loop simulation cost is not separable, long-term memory effects present due to small sample time

SOLUTION APPROACH

1. Standard-scale I/O data for numerical reasons $u_i \leftarrow \frac{u_i - \mu_u^i}{\sigma_u^i}, y_i \leftarrow \frac{y_i - \mu_y^i}{\sigma_y^i}$
 $i = 1, \dots, 6$

2. Train (A, B, C) by N4SID (Overschee, De Moor, 1994) with focus on simulation

See also results in recently updated report (Weigand, Götz, Ulmen, Ruskowski, 2023)

3. Train simple RESNET model with shallow NNs:

$$x_{k+1} = Ax_k + Bu_k + f_x(x_k, u_k, \theta_x), \quad y_k = Cx_k + f_y(x_k, \theta_y)$$

- **Optimization setup:** in Python, using **JAX** and **L-BFGS-B** (Byrd, Lu, Nocedal, Zhu, 1995) to handle ℓ_1 -regularization

TRAINING RNN W/ ℓ_1 -PENALTIES VIA L-BFGS-B

- To handle ℓ_1 -regularization, split $\theta_x = \theta_x^+ - \theta_x^-$ and $\theta_y = \theta_y^+ - \theta_y^-$:

$$\min_{\theta_x^+, \theta_y^+, \theta_x^-, \theta_y^-} \frac{1}{N} \sum_{k=1}^N \|y_k - \hat{y}_k\|_2^2 + \frac{1}{2} \rho \left\| \begin{bmatrix} \theta_x^+ \\ \theta_y^+ \\ \theta_x^- \\ \theta_y^- \end{bmatrix} \right\|_2^2 + \tau \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}' \begin{bmatrix} \theta_x^+ \\ \theta_y^+ \\ \theta_x^- \\ \theta_y^- \end{bmatrix}$$

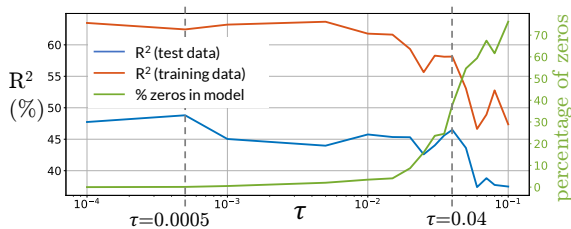
s.t. model equations, $x_0 = 0$

$$\theta_x^+, \theta_y^+, \theta_x^-, \theta_y^- \geq 0$$

- Lemma:** weighting $\|\theta_x^+\|_2^2 + \|\theta_x^-\|_2^2 + \|\theta_y^+\|_2^2 + \|\theta_y^-\|_2^2$ is equivalent to weighting $\|\theta_x^+ - \theta_x^-\|_2^2 + \|\theta_y^+ - \theta_y^-\|_2^2$ (proof is simple by contradiction)
- Note:** weighting $\|\theta_x^+\|_2^2 + \|\theta_x^-\|_2^2 + \|\theta_y^+\|_2^2 + \|\theta_y^-\|_2^2$ is **numerically better**, as ℓ_2 -regularization is strongly convex for $\rho > 0$

INDUSTRIAL ROBOT BENCHMARK: RESULTS

- State $x \in \mathbb{R}^{12}$, f_x, f_y with $n_1^x = 24$ and $n_1^y = 12$ neurons, respectively, $\rho = 10^{-4}$
- Total number of training parameters: $\dim(\theta_x) + \dim(\theta_y) = 990$



(best R^2 in 5 runs)

- Model quality measured by **average R^2 -score** on all outputs:

$$\overline{R^2} = \frac{1}{n_y} \sum_{i=1}^{n_y} 100 \left(1 - \frac{\sum_{k=1}^N (y_{k,i} - \hat{y}_{k,i|0})^2}{\sum_{k=1}^N (y_{k,i} - \frac{1}{N} \sum_{i=1}^N y_{k,i})^2} \right)$$

- Training time \approx **50 min** on a single core of an Apple M1 Max CPU

INDUSTRIAL ROBOT BENCHMARK: RESULTS

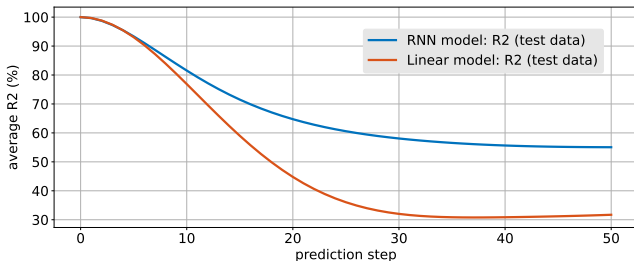
- **Open-loop simulation** errors ($\rho = 10^{-4}$, $\tau = 0.0005$, $n_1^x = 24$, $n_1^y = 12$):

	R^2 (training) RNN model	R^2 (test) RNN model	R^2 (training) linear model	R^2 (test) linear model
y_1	84.3099	74.3654	59.7335	59.9400
y_2	73.3438	53.2403	48.6032	31.9400
y_3	65.0838	47.0516	47.3231	24.1045
y_4	47.9524	46.2464	25.0829	21.6542
y_5	37.0665	34.3510	25.0987	24.8838
y_6	66.9417	37.5726	29.8516	31.5943
average	62.4497	48.8046	39.2822	32.3528

- **Note:** we tried different values of τ and number of neurons n_1^x, n_1^y :
max R^2 -score on test data = **48.9904** with $R^2 = 59.0654$ on training data
- More model parameters/smaller regularization leads to overfit training data

INDUSTRIAL ROBOT BENCHMARK: RESULTS

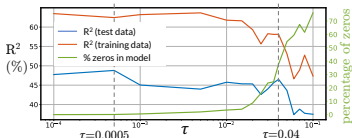
- Compute p -step ahead prediction $\hat{y}_{k+p|k}$, with hidden state $x_k|k$ estimated by an Extended Kalman Filter based on identified RNN model



- This is a more relevant indicator of model quality for MPC purposes than open-loop simulation error $\hat{y}_{k|0} - y_k$

INDUSTRIAL ROBOT BENCHMARK: RESULTS

- Compare **Adam** (Kingma, Ba, 2014) vs **L-BFGS-B**²:
($\tau = 0.04$, $\rho = 10^{-4}$, $n_1^x = 24$, $n_1^y = 12$)



method	best case criterion	average R^2 (training)	average R^2 (test)	# zeros	CPU time (s)
L-BFGS-B	R_2 (test)	58.13	46.49	375/990	3215
Adam		51.51	47.31	8/990	2511
L-BFGS-B	# zeros	54.34	45.07	520/990	3172
Adam		50.41	41.99	27/990	2518

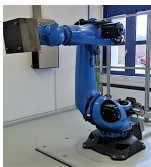
Adam: tuned with learning rate exponentially decaying from 0.01 after 1000 steps, with decay rate 0.05.

- L-BFGS-B leads to **sparser models** than Adam with similar R^2 -scores

²Best out of 5 runs, either based on the R_2 on test data or # zeros in θ_x, θ_y

CONCLUSIONS

- **Numerically challenging** nonlinear SYS-ID benchmark: long I/O sequences, many model parameters to learn
- Large training dataset, but easily overfit. **More experiments** required to be able to identify richer model structures
- Possible numerical improvements: **split data** into independent experiments, then **parallelize** cost/gradient computations
- A realistic physics-based **nonlinear robot simulation model** to assess model quality in terms of resulting **closed-loop control performance** would be great



I thank all contributors who made the dataset available!