

IN-CONTEXT LEARNING FOR MODEL-FREE SYSTEM IDENTIFICATION

Marco Forgone, Filippo Pura, Dario Piga

IDSIA Dalle Molle Institute for Artificial Intelligence SUPSI-USI, Lugano, Switzerland

September 7, 2023

Standard system identification/supervised machine learning

- 1 Collect dataset $\mathcal{D} = (u_{1:N}, y_{1:N})$ of input/outputs from system S .
- 2 Apply an algorithm to estimate a model $M(\hat{\theta})$ of S :

$$\hat{\theta} = \mathcal{A}(\mathcal{D}) \quad \text{e.g. } \mathcal{A}(\mathcal{D}) = \arg \min_{\theta \in \Theta} \mathcal{L}(\mathcal{D}, M(\theta))$$

- 3 Make predictions/simulations using the model on new data:

$$\hat{y}_{1:M}^* = M(u_{1:M}^*; \hat{\theta})$$

Researchers keep on improving learning algorithms and model structures.
Can we automate this process? Can we **learn the learning algorithm** itself?

Meta learning tries to answer this question.

 J. Schmidhuber. Evolutionary principles in self-referential learning, or on learning how to learn.
Diploma Thesis, TU Munich, 1987

 T. Hospedales et al. Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022*

Standard system identification/supervised machine learning

- 1 Collect dataset $\mathcal{D} = (u_{1:N}, y_{1:N})$ of input/outputs from system S .
- 2 Apply an algorithm to estimate a model $M(\hat{\theta})$ of S :

$$\hat{\theta} = \mathcal{A}(\mathcal{D}) \quad \text{e.g. } \mathcal{A}(\mathcal{D}) = \arg \min_{\theta \in \Theta} \mathcal{L}(\mathcal{D}, M(\theta))$$

- 3 Make predictions/simulations using the model on new data:

$$\hat{y}_{1:M}^* = M(u_{1:M}^*; \hat{\theta})$$

Researchers keep on improving learning algorithms and model structures.
Can we automate this process? Can we **learn the learning algorithm** itself?

Meta learning tries to answer this question.

 J. Schmidhuber. Evolutionary principles in self-referential learning, or on learning how to learn.
Diploma Thesis, TU Munich, 1987

 T. Hospedales et al. Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022*

Standard system identification/supervised machine learning

- 1 Collect dataset $\mathcal{D} = (u_{1:N}, y_{1:N})$ of input/outputs from system S .
- 2 Apply an algorithm to estimate a model $M(\hat{\theta})$ of S :


$$\hat{\theta} = \mathcal{A}(\mathcal{D}) \quad \text{e.g. } \mathcal{A}(\mathcal{D}) = \arg \min_{\theta \in \Theta} \mathcal{L}(\mathcal{D}, M(\theta))$$


- 3 Make predictions/simulations using the model on new data:

$$\hat{y}_{1:M}^* = M(u_{1:M}^*; \hat{\theta})$$

Researchers keep on improving learning algorithms and model structures.
Can we automate this process? Can we **learn the learning algorithm** itself?

Meta learning tries to answer this question.

 **J. Schmidhuber.** Evolutionary principles in self-referential learning, or on learning how to learn.
Diploma Thesis, TU Munich, 1987

 **T. Hospedales et al.** Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022

Our meta learning setting

- We have an **infinite stream** of datasets from a distribution $p(\mathcal{D})$:

$$\{\mathcal{D}^{(i)} = (u_{1:N}^{(i)}, y_{1:N}^{(i)}), i = 1, 2, \dots, \infty\}$$

- $\mathcal{D}^{(i)}$ generated by **random system** $S^{(i)}$ and input realization $u_{1:N}^{(i)}$
- Different but **related to each other**. There's a learnable structure!

Can we get better at identifying $S^{(i)}$ as we observe more datasets $\mathcal{D}^{(j)}$?

- $p(\mathcal{D})$ may be a **physical simulator** where we can change settings
- The learned algorithm could then be applied to **real data**

Meta learning from a **finite collection** would also be interesting...

Our meta learning setting

- We have an **infinite stream** of datasets from a distribution $p(\mathcal{D})$:

$$\{\mathcal{D}^{(i)} = (u_{1:N}^{(i)}, y_{1:N}^{(i)}), i = 1, 2, \dots, \infty\}$$

- $\mathcal{D}^{(i)}$ generated by **random system** $S^{(i)}$ and input realization $u_{1:N}^{(i)}$
- Different but **related to each other**. There's a learnable structure!

Can we get better at identifying $S^{(i)}$ as we observe more datasets $\mathcal{D}^{(j)}$?

- $p(\mathcal{D})$ may be a **physical simulator** where we can change settings
- The learned algorithm could then be applied to **real data**

Meta learning from a **finite collection** would also be interesting...

Our meta learning setting

- We have an **infinite stream** of datasets from a distribution $p(\mathcal{D})$:

$$\{\mathcal{D}^{(i)} = (u_{1:N}^{(i)}, y_{1:N}^{(i)}), i = 1, 2, \dots, \infty\}$$

- $\mathcal{D}^{(i)}$ generated by **random system** $S^{(i)}$ and input realization $u_{1:N}^{(i)}$
- Different but **related to each other**. There's a learnable structure!

Can we get better at identifying $S^{(i)}$ as we observe more datasets $\mathcal{D}^{(j)}$?

- $p(\mathcal{D})$ may be a **physical simulator** where we can change settings
- The learned algorithm could then be applied to **real data**

Meta learning from a **finite collection** would also be interesting...

Our meta learning setting

- We have an **infinite stream** of datasets from a distribution $p(\mathcal{D})$:

$$\{\mathcal{D}^{(i)} = (u_{1:N}^{(i)}, y_{1:N}^{(i)}), i = 1, 2, \dots, \infty\}$$

- $\mathcal{D}^{(i)}$ generated by **random system** $S^{(i)}$ and input realization $u_{1:N}^{(i)}$
- Different but **related to each other**. There's a learnable structure!

Can we get better at identifying $S^{(i)}$ as we observe more datasets $\mathcal{D}^{(j)}$?

- $p(\mathcal{D})$ may be a **physical simulator** where we can change settings
- The learned algorithm could then be applied to **real data**

Meta learning from a **finite collection** would also be interesting...

In-context learning

Many meta learning strategies around. Here focus on **in-context learning**.

- **Transformers** are expressive as a **programming language**
- We make Transformers **behave like algorithms**. We provide:
 - ▶ A **context**, namely an input/output sequence of a system
 - ▶ A **task**, like predicting the next output or simulating for more steps
- The Transformer must **learn to identify** the system to solve the task!

Context + task may be seen as a **prompt** to a Large Language Model, which can then **continue the word sequence** in an optimal way.



G. Weiss et al. Thinking like Transformers. *38th International Conference on Machine Learning (ICML 2022)*.



S. Garg et al. What Can Transformers Learn In-Context? A Case Study of Simple Function Classes. *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*.



T. Brown et al. Language Models are Few-Shot Learners. *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*.

Two system identification problems

One-step prediction:

$$\hat{y}_{k+1} = \mathcal{M}_\phi(u_{1:k}, y_{1:k})$$

- $(u_1, y_1) \rightarrow \hat{y}_2$
- $(u_{1:2}, y_{1:2}) \rightarrow \hat{y}_3$
- $(u_{1:3}, y_{1:3}) \rightarrow \hat{y}_4$
- ...
- $(u_{1:N-1}, y_{1:N-1}) \rightarrow \hat{y}_N$

Multi-step simulation

$$\hat{y}_{m+1:N} = \mathcal{M}_\phi(u_{1:m}, y_{1:m}, u_{m+1:N})$$

Meta-model receives:

- 1 Full input/output $(u_{1:m}, y_{1:m})$
 - 2 Input-only trajectory $u_{m+1:N}$
- and generates simulation: $\hat{y}_{m+1:N}$

- $u_{1:m}, y_{1:m}, u_{m+1:N} \rightarrow \hat{y}_{m+1:N}$

- If we manage to train a Transformer \mathcal{M}_ϕ to solve such problems for a class of systems, it becomes a meta model of that class!
- \mathcal{M}_ϕ becomes as powerful as a system identification algorithm!

Two system identification problems

One-step prediction:

$$\hat{y}_{k+1} = \mathcal{M}_\phi(u_{1:k}, y_{1:k})$$

- $(u_1, y_1) \rightarrow \hat{y}_2$
- $(u_{1:2}, y_{1:2}) \rightarrow \hat{y}_3$
- $(u_{1:3}, y_{1:3}) \rightarrow \hat{y}_4$
- ...
- $(u_{1:N-1}, y_{1:N-1}) \rightarrow \hat{y}_N$

Multi-step simulation

$$\hat{y}_{m+1:N} = \mathcal{M}_\phi(u_{1:m}, y_{1:m}, u_{m+1:N})$$

Meta-model receives:

- ① Full input/output $(u_{1:m}, y_{1:m})$
 - ② Input-only trajectory $u_{m+1:N}$
- and generates simulation: $\hat{y}_{m+1:N}$
- $u_{1:m}, y_{1:m}, u_{m+1:N} \rightarrow \hat{y}_{m+1:N}$

- If we manage to train a Transformer \mathcal{M}_ϕ to solve such problems for a class of systems, it becomes a meta model of that class!
- \mathcal{M}_ϕ becomes as powerful as a system identification algorithm!

Meta model training

One-step prediction:

$$\hat{\phi} = \arg \min_{\phi} \mathcal{L}_{\text{pred}}(\phi)$$

$$\mathcal{L}_{\text{pred}}(\phi) = \mathbb{E}_{p(\mathcal{D})} \left[\sum_{k=1}^{N-1} \|y_{k+1} - \mathcal{M}_{\phi}(y_{1:k}, u_{1:k})\|^2 \right]$$

$$\mathcal{L}_{\text{pred}}(\phi) \approx \frac{1}{b} \sum_{i=1}^b \sum_{k=1}^{N-1} \|y_{k+1}^{(i)} - \mathcal{M}_{\phi}(y_{1:k}^{(i)}, u_{1:k}^{(i)})\|^2$$

Multi-step simulation

$$\hat{\phi} = \arg \min_{\phi} \mathcal{L}_{\text{sim}}(\phi)$$

$$\mathcal{L}_{\text{sim}}(\phi) = \mathbb{E}_{p(\mathcal{D})} \left[\|y_{m+1:N} - \mathcal{M}_{\phi}(u_{1:m}, y_{1:m}, u_{m+1:N})\|^2 \right]$$

$$\mathcal{L}_{\text{sim}}(\phi) \approx \frac{1}{b} \sum_{i=1}^b \|y_{m+1:N}^{(i)} - \mathcal{M}_{\phi}(u_{1:m}^{(i)}, y_{1:m}^{(i)}, u_{m+1:N}^{(i)})\|^2$$

- Formally, just two boring supervised learning problems.
- The use of powerful architectures and training on a whole class of dynamical systems makes the outcome special.
- If the optimization works out well, the Transformer becomes a meta model of the systems in $p(\mathcal{D})$!
- We have learned a learning algorithm!

Meta model training

One-step prediction:

$$\hat{\phi} = \arg \min_{\phi} \mathcal{L}_{\text{pred}}(\phi)$$

$$\mathcal{L}_{\text{pred}}(\phi) = \mathbb{E}_{p(\mathcal{D})} \left[\sum_{k=1}^{N-1} \|y_{k+1} - \mathcal{M}_{\phi}(y_{1:k}, u_{1:k})\|^2 \right]$$

$$\mathcal{L}_{\text{pred}}(\phi) \approx \frac{1}{b} \sum_{i=1}^b \sum_{k=1}^{N-1} \|y_{k+1}^{(i)} - \mathcal{M}_{\phi}(y_{1:k}^{(i)}, u_{1:k}^{(i)})\|^2$$

Multi-step simulation

$$\hat{\phi} = \arg \min_{\phi} \mathcal{L}_{\text{sim}}(\phi)$$

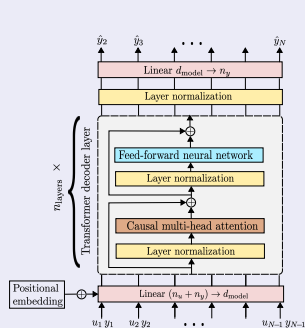
$$\mathcal{L}_{\text{sim}}(\phi) = \mathbb{E}_{p(\mathcal{D})} \left[\|y_{m+1:N} - \mathcal{M}_{\phi}(u_{1:m}, y_{1:m}, u_{m+1:N})\|^2 \right]$$

$$\mathcal{L}_{\text{sim}}(\phi) \approx \frac{1}{b} \sum_{i=1}^b \|y_{m+1:N}^{(i)} - \mathcal{M}_{\phi}(u_{1:m}^{(i)}, y_{1:m}^{(i)}, u_{m+1:N}^{(i)})\|^2$$

- Formally, just two **boring supervised learning** problems.
- The use of powerful architectures and **training on a whole class** of dynamical systems makes the outcome special.
- If the optimization works out well, the Transformer becomes a **meta model** of the systems in $p(\mathcal{D})$!
- We have **learned a learning algorithm**!

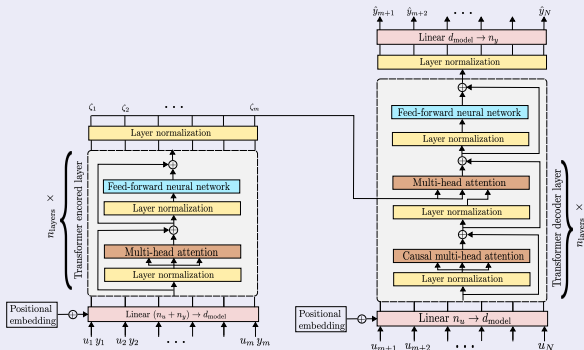
Transformer architectures

One-step prediction:



decoder-only (\sim GPT-2)

Multi-step simulation



encoder-decoder (\sim language translation)

NLP architectures modified to process **real-valued** input/output sequences.

Experiments - System classes

One-step prediction and multi-step simulation on two system classes:

Linear Time Invariant (LTI):

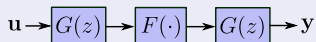
In state-space form, order ≤ 10

$$x_{k+1} = Ax_k + Bu_k$$

$$y_{k+1} = Cx_k$$

- Random system matrices
- A constrained to be stable

Wiener-Hammerstein (WH):



- Sequential LTI $\rightarrow F(\cdot) \rightarrow$ LTI
- Random LTI, order ≤ 5
- $F(\cdot)$: random feedforward NN.

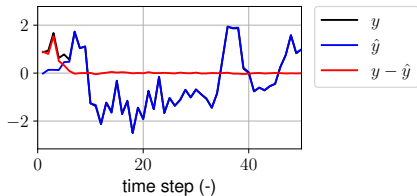
- For both classes, input $u_{1:N}$ is a white Gaussian noise sequence.
- This defines a $p(\mathcal{D})$. We can generate infinite datasets!
- Each dataset from a different input/system realization!



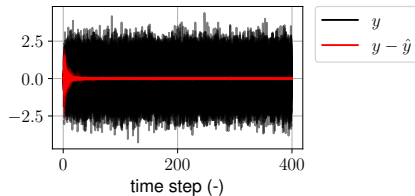
<https://github.com/forgi86/sysid-transformers>

Experiments - one-step prediction results

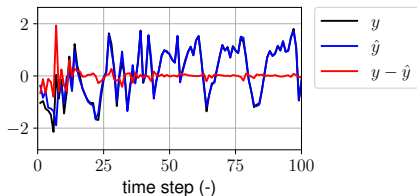
LTI: one sequence



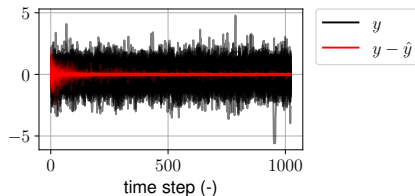
LTI: 256 sequences



WH: one sequence

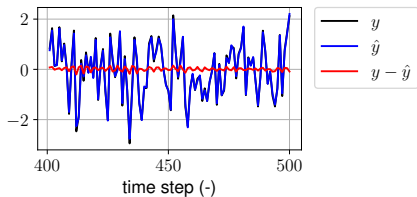


WH: 32 sequences

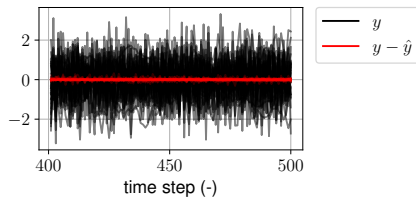


Experiments - multi-step simulation results

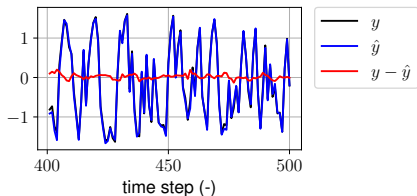
LTI: one sequence



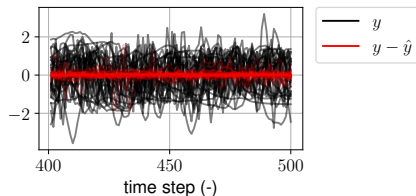
LTI: 256 sequences



WH: one sequence



WH: 32 sequences



Conclusions

An **in-context** learning approach for system identification.

- Model-free, no need to re-train for a specific dataset/system
- Exploits the power of Transformers seen as **trainable algorithms**
- Seems to work!

Many possible research directions including:

- Transfer learning from one system class to another...
- ... and from simulation to reality.

Conclusions

An **in-context** learning approach for system identification.

- Model-free, no need to re-train for a specific dataset/system
- Exploits the power of Transformers seen as **trainable algorithms**
- Seems to work!

Many possible research directions including:

- Transfer learning from one system class to another...
- ... and from simulation to reality.

Thank you.
Questions?

`marco.forgione@idsia.ch`