Learning Equations for Extrapolation and Control

Subham Sekhar Sahoo[1,3], Christoph Lampert[2], and Georg Martius[1]

[1] Max Planck Institute for Intelligent Systems, Tuebingen, Germany [2] IST Austria, Klosterneuburg, Austria [3] IIT, Kharagpur, India

georg.martius@tuebingen.mpg.de

Abstract

In classical machine learning, regression is treated as a black box process of identifying a suitable function from a hypothesis set without attempting to gain insight into the mechanism connecting inputs and outputs. In the natural sciences, however, finding an interpretable function for a phenomenon is the prime goal as it allows to understand and generalize results. This paper proposes a novel type of function learning network, called **equation learner (EQL$^\div$)**, that can learn analytical expressions and is able to extrapolate to unseen domains. It is implemented as an end-to-end differentiable feed-forward network and allows for efficient gradient based training. Due to sparsity regularization concise interpretable expressions can be obtained. Applied to robot control, we can identify the dynamics equations after 2 random trials good enough to control a cart-pendulum to swing up and balance.

At a glance

What: finding the simplest descriptive formula for data

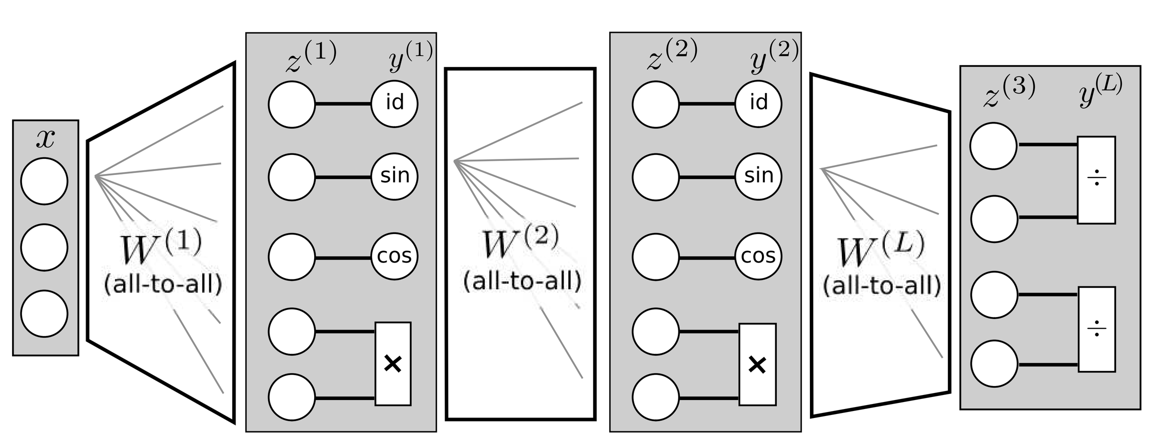
Why:
- to extrapolate to new situations,
- to dissect outcomes into causal pathways,
- to be efficient on evaluation

Example: a robot can make predictions about movements outside the experienced domain, e. g. for higher velocities.

How: differentiable network with analytic base functions, sparsity regularization and special model selection.

Network for function extrapolation

Architecture



Network architecture of the proposed Equation Learner with divisions (EQL$^\div$) for 3 layers ($L = 3$) and one neuron per type.

Each layer has:
- a linear all-to-all mapping to an intermediate representation z
- unary units implementing: identity , sine, and cosine
- binary units: multiplication of two inputs

The final layer computes the regression values as division.



Georg Martius

<georg.martius@tuebingen.mpg.de>

Training

Objective

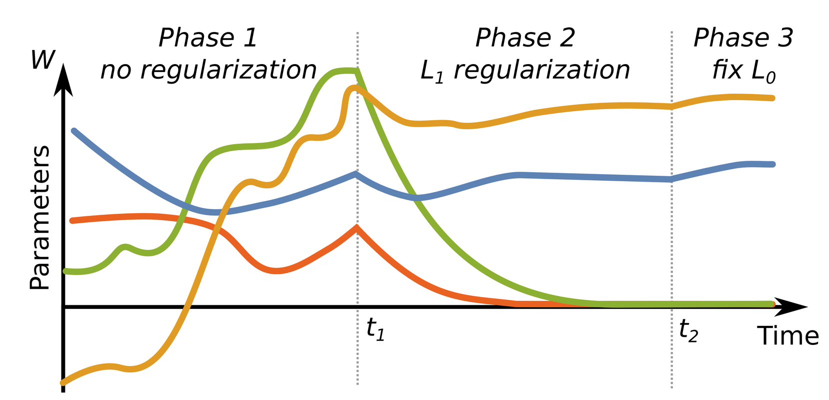
We use a Lasso-like objective (L_2 loss and L_1 regularization):

$$\mathcal{L}(D) = \frac{1}{N}\sum_{i=1}^{|D|}\|\psi(x_i)-y_i\|^2 + \lambda\sum_{l=1}^{L}\left|W^{(l)}\right|_1,$$

with network $\psi(x)$ and apply a stochastic gradient descent (Adam [1]) with mini-batches.

Learning/Regularization stages

Training is split into phases, because:
- plain L_1 regularization leads often to premature convergence to suboptima
- result is always trade-off between error and regularization term



- Phase 1: no regularzation ($\lambda = 0$)
- Phase 2: L_1 regularization ($\lambda > 0$)
- Phase 3: limit L_0 norm: $\{w = 0 \mid |w| < 0.001, w \in W^{1\ldots L}\}$

Division is regularized: one-sided and cut-off threshold $\theta = 1/\sqrt{t}$

$$h^\theta(a,b) := \begin{cases}\frac{a}{b} & \text{if } b > \theta\\ 0 & \text{otherwise}\end{cases}$$

Model selection

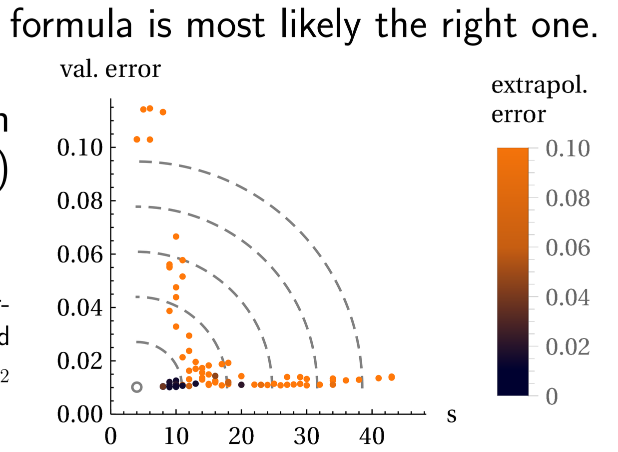
How to find the "right" formula?

1) Without any data from extrapolation domain:

Occams razor: the **simplest** formula is most likely the right one.

$\Longrightarrow$ Pick the instance with lowest complexity (# units) and lowest validation error

kin-4-end dataset: extrapolation performance depending on validation error and sparsity s. Circle arcs indicate the L_2 norm iso-lines (normalized).

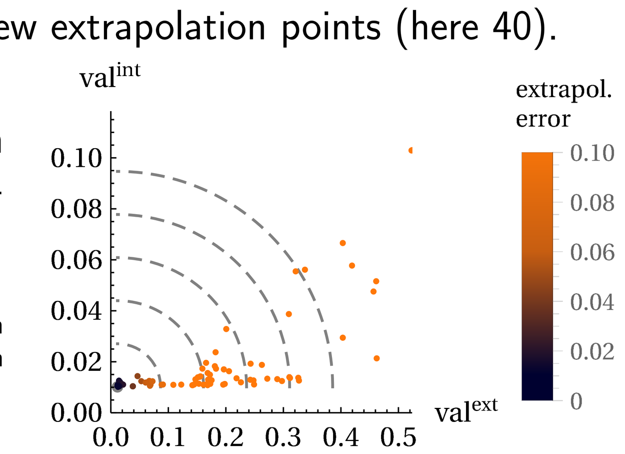


2) With some points from extrapolation domain:

Use also validation error on few extrapolation points (here 40).

$\Longrightarrow$ Pick instance with lowest validation in interpolation and extrapolation

as above but using validation error in both domains. Circle arcs indicate the L_2 norm iso-lines (normalized).

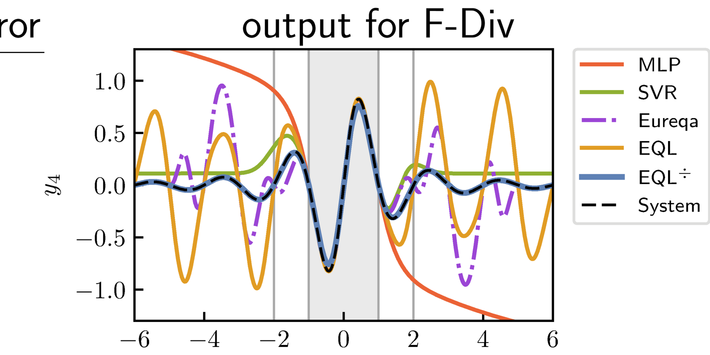


Results on complex formula

Formula containing a division:

$$y = \frac{\sin(\pi x_1)}{(x_2^2+1)}$$ F-Div

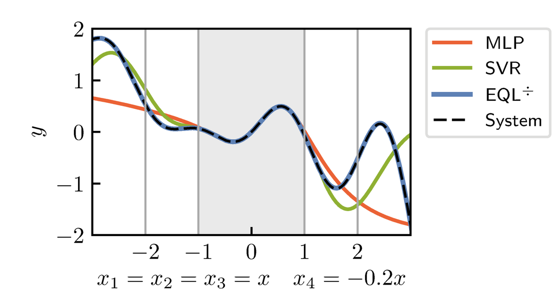
	extrapolation error
EQL$^\div$	0.01
EQL	0.20
MLP	0.83
SVR	0.26
Eureqa	0.13



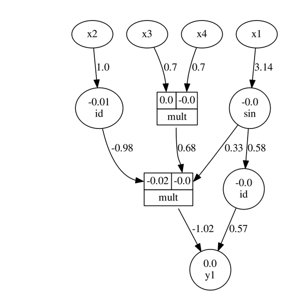
Consider data from more complicated formulas:

$$y = 1/3\left((1+x_2)\sin(\pi x_1) + x_2x_3x_4\right) \quad \text{F-3}$$
$$y = 1/2\left(\sin(\pi x_1) + \cos(2x_2\sin(\pi x_1)) + x_2x_3x_4\right) \quad \text{F-4}$$

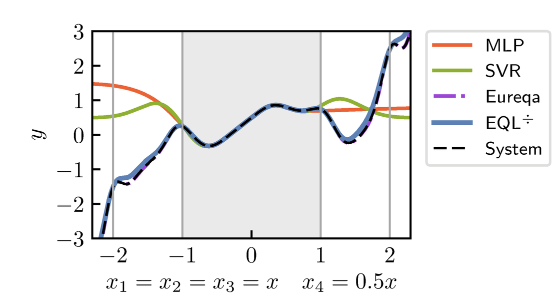
F-3



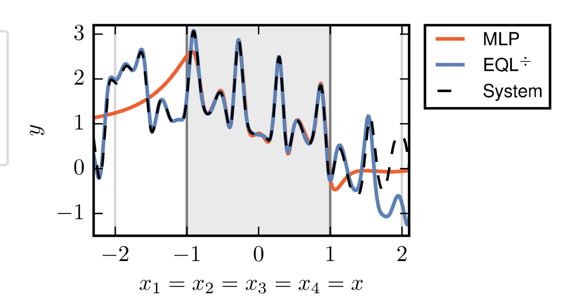
learned equation graphs



F-4



random expression



training domain 1/16th of extrapolation domain!

extrapolation	EQL$^\div$	EQL	MLP	SVR	Eureqa
F-3	0.01	0.35	0.47	0.34	0.01
F-4	0.23	0.37	0.86	0.91	0.85
Random Exp	0.03	—	1.89	17.67	—

There are also cases when we cannot find the right formula, see [4]

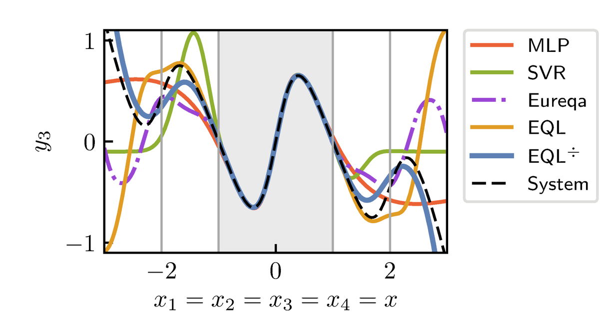
Cart-pendulum dynamics

Learn dynamics equation from synthetic data

Forward dynamics contains divisions: ($y_3 = \dot{\theta}$)

$$y_3 = \frac{-x_1 - 0.01x_3 + x_4^2\sin(x_2) + 0.1x_4\cos(x_2) + 9.81\sin(x_2)\cos(x_2)}{\sin^2(x_2)+1}$$

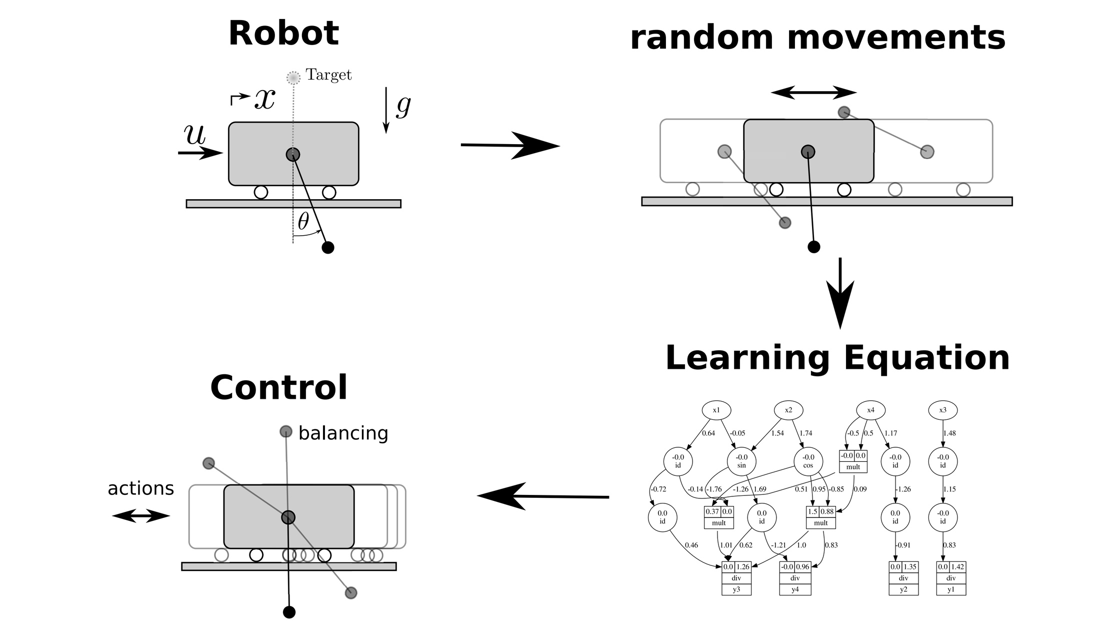
Equations of motion randomly sampled from subdomain ($[-1, 1]$)



	interp.	extrapol.
EQL$^\div$	0.010	0.06
EQL	0.0103	0.17
MLP	0.012	0.18
SVR	0.019	0.36
Eureqa	0.012	0.19

Great extrapolation, but needs to be realizable!

Learning to control cart-pendulum

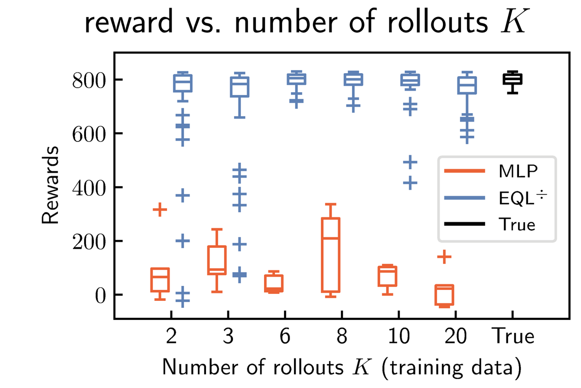


- modified OpenAI Gym cart-pole for swingup
- collect data from K random rollouts
- train EQL$^\div$ networks on $K-1$ rollouts from scratch
- use one for validation $\Longrightarrow$ find best equation
- use model predictive control (MPC) to perform cart-pendulum swingup

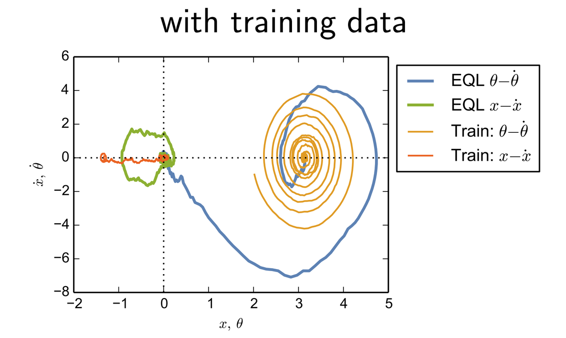
Utility function for MPC: (pole up and cart in the center)

$$R = -\cos(\theta) + 0.1x^2 + 0.1\dot{x}^2 + 0.02\dot{\theta}^2$$

reward vs. number of rollouts K



MPC on EQL$^\div$ model (with K=2) with training data



Successful swingup after 2 random trails!

Video: `https://youtu.be/MG9q3gTtBLs`

Conclusion

- *Equation Learner* (EQL$^\div$) learns analytical expressions from data with divisions
- symbolic regression as continuous optimization problem
- special model selection procedure following Occams razor
- works for a wide range of examples
- suitable for dynamics learning and control: cartpole swingup after 2 random rollouts

Code: `https://github.com/martius-lab/EQL`

References

[1] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *in Proceedings of ICLR*, 2015.

[2] Georg Martius and Christoph H. Lampert. Extrapolation and learning equations. *arXiv:1610.02995*, 2016.

[3] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[4] Subham S. Sahoo, Christoph H. Lampert and Georg Martius. Learning equations for extrapolation and control. *ICML 2018*, Stockholm, Sweden, 2018