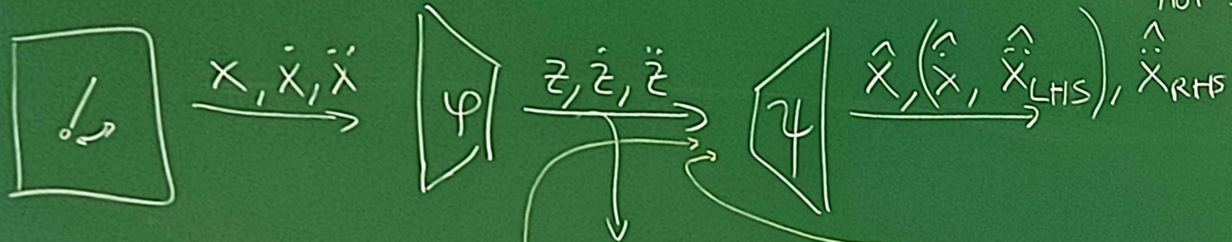
Correction. Chain-Rule Algorithm

2 c $\ddot{z}^{l} \leftarrow \left(g''(\tilde{z}^{l}) \odot \dot{\tilde{z}}^{l}\right) \odot \dot{\tilde{z}}^{l} + \left(g'(\tilde{z}^{l}) \odot \ddot{\tilde{z}}^{l}\right)$

not squared!



$$\underbrace{\ddot{z}}_{LHS} \approx \underbrace{\Theta(z, \dot{z}) \boxed{\;-\;}}_{RHS} = \hat{\ddot{z}}$$

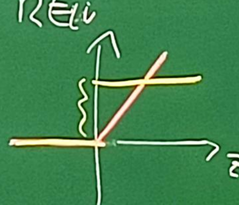
ideally equal
at the end of training

Activation Functions

Sigmoid: $\sigma(z) = \frac{1}{1+e^{-x}}$

$$\sigma'(z) = \sigma(z) \cdot (1-\sigma(z))$$

$$\sigma''(z) = \sigma'(z) \cdot (1-2\sigma(z))$$



Also: - RELU (careful: discontinuous $RELU'(z)$)

- ELU, GELU
- Dropout (careful: z becomes noisy)
- more complicated g: BatchNorm (careful: complicated g', g'')

→ analytically computing the derivatives may require some prior work but it is computationally very efficient

Alternatives : - Autograd : very expensive

- Numerical differentiation : very noisy

Training

- Two components : Autoencoder , SINDy coefficient

Why do we train jointly? → SINDy encourages the encoder to learn suitable low-dimensional coordinates in which the equation can be formulated from the pre-defined terms

Loss

Reconstruction term $\mathcal{L}_x$

- x
- $x \xrightarrow{\varphi} z \xrightarrow{\psi} \hat{x}$

$$\rightarrow \mathcal{L}_x = \| x - \underbrace{\psi(\varphi(x))}_{\hat{x}} \|_2^2$$

Dynamics term in z : $\mathcal{L}_{\dot{z}}$ (for 1st order ODE for simplicity: $\dot{z} = f(z)$)

"How good does the LHS and RHS of the equation match?"

- $\dot{x} \xrightarrow{\varphi} \dot{z}$
- $x \xrightarrow{\varphi} z \xrightarrow{\Theta(z)\Xi} \hat{\dot{z}}$

$\rightarrow \mathcal{L}_{\dot{z}} = \| \underbrace{\nabla_x \varphi(x)\dot{x}}_{\text{LHS}, \dot{z}} - \underbrace{\Theta(\varphi(x))\Xi}_{\text{RHS}, \hat{\dot{z}}} \|_2^2$

via encoder chain rule

$$\left[\begin{array}{l} \text{1st order ODE: } \dot{z} = f(z) = \Theta(z)\Xi \\ \text{2nd order ODE: } \ddot{z} = f(z,\dot{z}) = \Theta(z,\dot{z})\Xi \end{array}\right]$$

Dynamics term in x : $\mathcal{L}_{\dot{x}}$

"How close to $\dot{x}$ is the decoded RHS?"

- $\dot{x}$
- $x \xrightarrow{\varphi} z \xrightarrow{\Theta(z)\Xi} \hat{\dot{z}}_{RHS} \xrightarrow{\psi} \hat{\dot{x}}_{RHS}$

$\rightarrow \mathcal{L}_{\dot{x}} = \| \dot{x} - \underbrace{\nabla_z \psi(\varphi(x))}_{\text{decoder chain rule}} \cdot \underbrace{\Theta(\varphi(x))\Xi}_{\text{RHS}, \hat{\dot{z}}} \|_2^2$

LASSO Regularization of Ξ

$$\mathcal{L}_1 = \|\Xi\|_1$$

$$\hat{\varphi}, \hat{\psi}, \hat{\Xi} = \underset{\varphi, \psi, \Xi}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \left[\mathcal{L}_x^{(i)} + \lambda_{\dot{z}} \mathcal{L}_{\dot{z}}^{(i)} + \lambda_{\dot{x}} \mathcal{L}_{\dot{x}}^{(i)} \right] + \lambda_1 \mathcal{L}_1$$

Loss for 2nd order ODE:

- Replace $\dot{z}, \dot{x} \rightarrow \ddot{z}, \ddot{x}$, no loss for intermediate (1st order) time derivatives

Thresholding: Same as in SINDy on z

<u>Refinement</u>

- Due to the L_1 loss, the coefficients are biased towards smaller values and do not result in the optimal equation for the dynamics

- After convergence with $\lambda_1 \to 0$ and thresholding of unimportant coefficients, proceed training without L_1 loss ($\lambda_1 = 0$) an let the model "refine" its coordinates and the equation.

- Once a coefficient is thresholded / disabled during training, it remains 0 for the rest of further training, including refinement

Caveats

- Sequential Thresholding does not work well → not very robust / reliable
- Requirement of prior knowledge about the system (Need to guess plausible terms for SINDy before training
  → also for $\dim(z)$ → choose terms for $\Theta = [1, z, \dot{z}, \sin(z), \dots]$
- Inherent Freedom of Choice : for a given dynamic system, many coordinate systems may be valid to describe the dynamics

Balance:

- few, simple terms: coordinates are also shaped simple s.t. they fit the equation
- many/complicated terms: Autoencoder may find one of many coordinates in which the on equation can be constructed from the terms ($\longrightarrow$ unidentifiability)

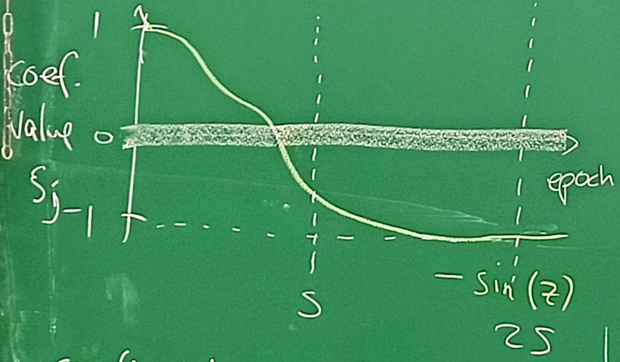
$\rightarrow$ "hallucinations": terms that correspond to non-existent effects of the ground truth system

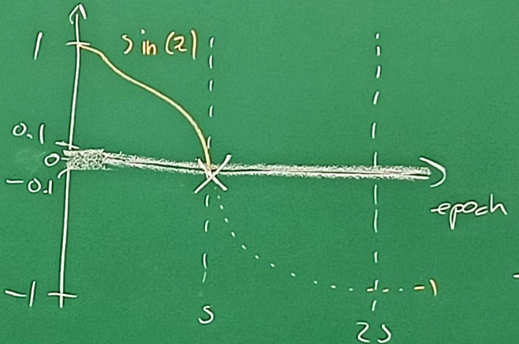
GT: $\ddot{z} = -\sin(z)$

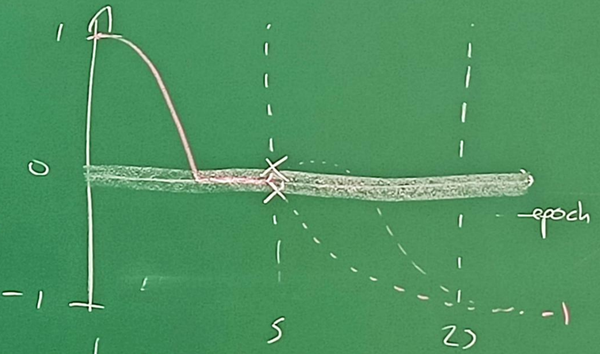
Sometimes: $\ddot{z} = 0.5 - 0.9\sin(z)$

possible reason: coordinates are not the same as the ground truth coordinates

Problems with Sequential Thresholding [Saegert 2022] @psaegert







coeficient pass trough the threshold region fast enough and reach $-\sin(z)$

coeficient happens to pass through the treshold region exactly at the threshold epoch $n \cdot S$ $n \in \mathbb{N}$ and are turned off although they are still changing and not converged

all coeficients quickly drop to 0 (possibly due to inconvenient initialization of the Autoencoder

→ messy coordinates in which none of the coeficients can contribute to solving the equation

→ before the coordinates can change to more suitable ones, all coef. are turned off.

→ most equations with ST end up being $\vec{\xi} = 0$. only occasionally it discovers the "right" or ground truth equation (due to random initialization)

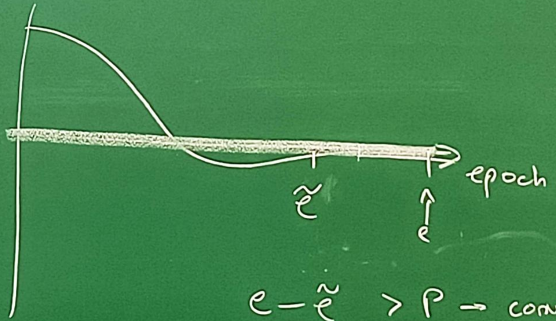
→ still need thresholding for sparsity

Patient Trend-Aware Thresholding (PTAT)

In each epoch e, turn off coefficients, only if $\forall e' : e - P < e' \leq e$

1. $|\xi_{e'}| < a$
2. $|\xi_{e'} - \xi_{e'-1}| < b$

with hyperparameters a, b, P (ST: only have a)



$e - \tilde{e} > P$ → converged

cont.

→ Fewer false negatives : coefficients are allowed to evolve and converge to a possibly non-zero value

<u>Results</u> : (N = 10 random initializations)

<u>Sequential Thresholding</u>	<u>Patient Trend-Aware Thresholding</u>
35% : $\ddot{z} = 0$	0% : $\ddot{z} = 0$
25% : $\ddot{z} = f(z, \dot{z})$, no $\sin(z)$	0% : $\ddot{z} = f(z, \dot{z})$, no $\sin(z)$
40% : $\ddot{z} = -\sin(z) + f(z, \dot{z})$	100% : $\ddot{z} = -\sin(z) + f(z, \dot{z})$ 10/10

→ slightly more terms than ST, but more reliable and honest