Neural Power Units

Arithmetic extrapolation with fractional powers
NeurIPS | 2020

Niklas Heim, Václav Šmídl, Tomáš Pevný

Artificial Intelligence Center, Czech Technical University

`https://github.com/nmheim/NeuralArithmetic.jl`

Arithmetic extrapolation

→Neural Networks are great at interpolation, but they poorly extrapolate.

→Neural arithmetic assumes that the problem is composed of arithmetic operations.

Examples

Function approximation

$$f(x,y) = (x+y,\ xy,\ \frac{x}{y},\ \sqrt{x}\ )^T$$

Differential equations in physical / financial modelling

$$\begin{bmatrix} \dot{S} \\ \dot{I} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} -\beta & 0 & \eta \\ \beta & -\alpha & 0 \\ 0 & \alpha & \eta \end{bmatrix} \begin{bmatrix} I^\gamma S^\kappa \\ I \\ R \end{bmatrix}$$

NALU - Trask et al.

Definition: Neural Arithmetic Logic Unit (NALU)

$$\text{Addition: } \boldsymbol{a} = \hat{\boldsymbol{W}}\boldsymbol{x} \qquad \hat{\boldsymbol{W}} = \tanh(\boldsymbol{W}) \odot \sigma(\boldsymbol{M})$$

$$\text{Multiplication: } \boldsymbol{m} = \exp(\hat{\boldsymbol{W}} \log(|\boldsymbol{x}| + \epsilon))$$

$$\text{Output: } \boldsymbol{y} = \boldsymbol{a} \odot \boldsymbol{g} + \boldsymbol{m} \odot (1 - \boldsymbol{g}) \qquad \boldsymbol{g} = \sigma(\boldsymbol{G}\boldsymbol{x})$$

NALU has constrained weights, and is gating between an addition ($+$) and multiplication/division ($\times, \div$) path.

Inconsistent convergence, negative numbers not handled correctly.

NMU & NAU - Madsen & Johansen

Definition: Neural Multiplication & Addition Units

$$\text{NMU: } y_j = \prod_i \hat{M}_{ij} x_i + 1 - \hat{M}_{ij} \qquad \hat{M}_{ij} = \min(\max(M_{ij}, 0), 1) \tag{1}$$

$$\text{NAU: } \boldsymbol{y} = \hat{\boldsymbol{A}}\boldsymbol{x} \qquad \hat{A}_{ij} = \min(\max(A_{ij}, -1), 1) \tag{2}$$

NMU & NAU have constrained weights, and are learning $(+)$ and $(\times)$ by stacking.

No division $(\div)$.

Neural Power Unit

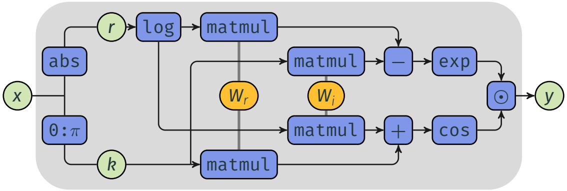
We improved NALU's multiplication path

$$m = \exp(W \log_{\text{real}}(|x|))$$

by lifting it into complex space $(\log := \log_{\text{complex}})$

$$z = \text{Re}(\exp(W_{\mathbb{C}} \log x)) = \text{Re}(\exp((W_r + iW_i) \log x)).$$

Naive Neural Power Unit (NaiveNPU)

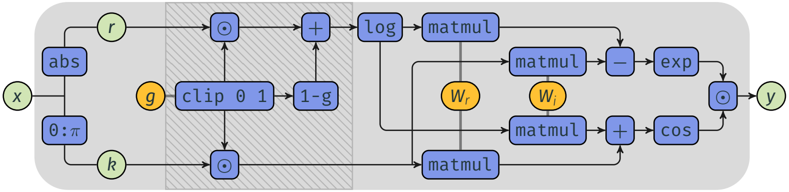


Definition: *Naive* Neural Power Unit

$$\boldsymbol{y} = \exp(\boldsymbol{W}_r \log \boldsymbol{r} - \pi \boldsymbol{W}_i \boldsymbol{k}) \odot \cos(\boldsymbol{W}_i \log \boldsymbol{r} + \pi \boldsymbol{W}_r \boldsymbol{k}), \text{ where}$$

$$\boldsymbol{r} = |\boldsymbol{x}| + \epsilon, \quad k_i = \begin{cases} 0 & x_i \geq 0 \\ 1 & x_i < 0 \end{cases},$$

Neural Power Unit (NPU) & Relevance Gate



Definition: Neural Power Unit

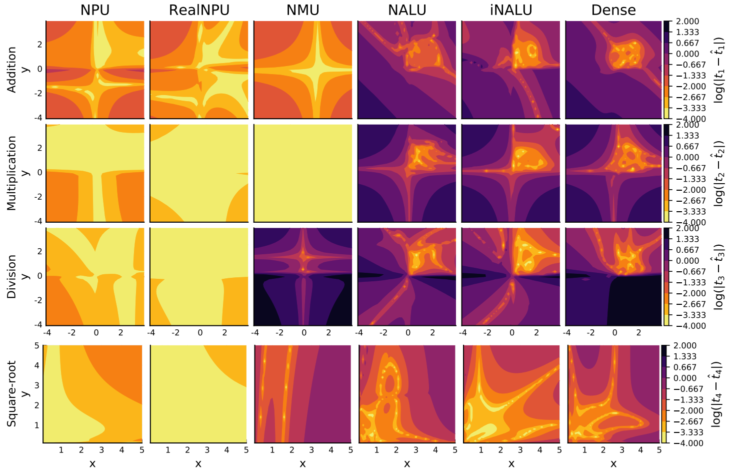
$$y = \exp(W_r \log r - \pi W_i k) \odot \cos(W_i \log r + \pi W_r k), \text{ where}$$

$$r = \hat{g} \odot (|x| + \epsilon) + (1 - \hat{g}),$$

$$k_i = \begin{cases} 0 & x_i \geq 0 \\ \hat{g}_i & x_i < 0 \end{cases}, \quad \hat{g}_i = \min(\max(g_i, 0), 1),$$

Learning Simple Arithmetic

$$f(x, y) = \left(x + y,\ xy,\ \frac{x}{y},\ \sqrt{x}\ \right)^T$$

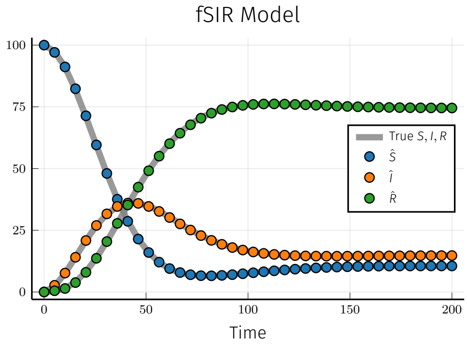


RealNPU denotes the NPU with $W_i = 0$.

Towards Equation Discovery

The fractional SIR model (fSIR, Taghvaei et al. [2020])

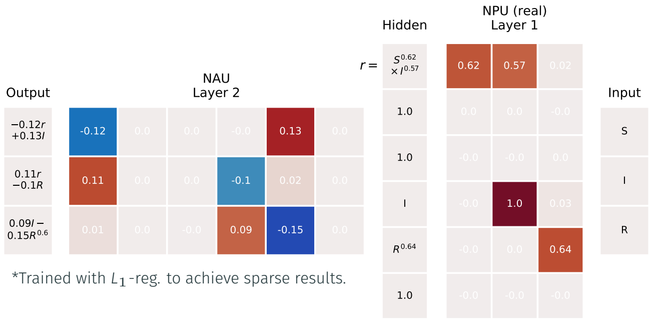
$$\begin{bmatrix} \dot{S} \\ \dot{I} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} -\beta & 0 & \eta \\ \beta & -\alpha & 0 \\ 0 & \alpha & \eta \end{bmatrix} \begin{bmatrix} I^{\gamma} S^{\kappa} \\ I \\ R \end{bmatrix}, \quad \begin{matrix} \alpha = 0.05 \\ \beta = 0.05 \\ \eta = 0.01 \\ \gamma = \kappa = 0.5 \end{matrix}$$



Towards Equation Discovery

The fractional SIR model (fSIR, Taghvaei et al. [2020])

$$\begin{bmatrix} \dot{S} \\ \dot{I} \\ \dot{R} \end{bmatrix} = \begin{bmatrix} -\beta & 0 & \eta \\ \beta & -\alpha & 0 \\ 0 & \alpha & \eta \end{bmatrix} \begin{bmatrix} I^\gamma S^\kappa \\ I \\ R \end{bmatrix}, \quad \begin{aligned} \alpha &= 0.05 \\ \beta &= 0.05 \\ \eta &= 0.01 \\ \gamma = \kappa &= 0.5 \end{aligned}$$



*Trained with L_1-reg. to achieve sparse results.

Neural Power Units

Arithmetic extrapolation with fractional powers
NeurIPS | 2020

Niklas Heim, Václav Šmídl, Tomáš Pevný

Artificial Intelligence Center, Czech Technical University

`https://github.com/nmheim/NeuralArithmetic.jl`