Package ‘nnR’

February 14, 2024

Type Package

Title Neural Networks Made Algebraic

Version 0.1.0

Maintainer Shakil Rafi <sarafi@uark.edu>

Description Do algebraic operations on neural networks. We seek here to implement in R, operations on neural networks and their resulting approximations. Our operations derive their descriptions mainly from
Rafi S., Padgett, J.L., and Nakarmi, U. (2024), ``Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials'', <doi:10.48550/arXiv.2402.01058>,
Grohs P., Hornung, F., Jentzen, A. et al. (2023), ``Space-time error estimates for deep neural network approximations for differential equations'', <doi:10.1007/s10444-022-09970-2>,
Jentzen A., Kuckuck B., von Wurstemberger, P. (2023), ``Mathematical Introduction to Deep Learning Methods, Implementations, and Theory'' <doi:10.48550/arXiv.2310.20360>.
Our implementation is meant mainly as a pedagogical tool, and proof of concept. Faster implementations with
deeper vectorizations may be made in future versions.

License GPL-3

Encoding UTF-8

Depends R (>= 4.1.0)

RoxygenNote 7.3.0

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/2shakilrafi/nnR/

BugReports https://github.com/2shakilrafi/nnR/issues?q=is%3Aissue+is%3Aopen+sort%3Aupdated-desc

VignetteBuilder knitr

NeedsCompilation no

Author Shakil Rafi [aut, cre] (<https://orcid.org/0000-0003-3791-9697>),
Joshua Lee Padgett [aut] (<https://orcid.org/0000-0001-9369-351X>),
Ukash Nakarmi [ctb] (<https://orcid.org/0000-0002-5351-3956>)

Repository CRAN

Date/Publication 2024-02-14 21:30:02 UTC

R topics documented:

`A` *This is an intermediate variable. See the reference*

Description

This is an intermediate variable. See the reference

Usage

```
A
```

Format

An object of class `matrix` (inherits from `array`) with 4 rows and 1 columns.

References

Definition 2.22. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

`Aff` *Aff*

Description

The function that returns Aff neural networks.

Usage

```
Aff(W, b)
```

Arguments

`W`	An $m \times n$ matrix representing the weight of the affine neural network
`b`	An $m \times 1$ vector representing the bias of the affine neural network

Value

Returns the network $((W, b))$ representing an affine neural network. Also denoted as $\mathsf{Aff}_{W,b}$ See also `Cpy` and `Sum`.

References

Definition 2.3.1. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360

And especially:

Definition 2.8. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Aff(4, 5)
c(5, 6, 7, 8, 9, 10) |>
  matrix(2, 3) |>
  Aff(5)
```

A_k   *A_k: The function that returns the matrix A_k*

Description

A_k: The function that returns the matrix A_k

Usage

```
A_k(k)
```

Arguments

k	Natural number, the precision with which to approximate squares within $[0, 1]$

Value

An intermediate matrix in a neural network that approximates the square of any real within $[0, 1]$ upon ReLU instantiation.

References

Definition 2.22. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
A_k(4)
A_k(45)
```

`B` *This is an intermediate variable, see reference.*

Description

This is an intermediate variable, see reference.

Usage

```
B
```

Format

An object of class `matrix` (inherits from `array`) with 4 rows and 1 columns.

`ck` *The ck function*

Description

The ck function

Usage

```
ck(k)
```

Arguments

`k`	input value, any real number

Value

the ck function

References

Definition 2.22. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
ck(1)
ck(-1)
```

comp *comp*

Description

The function that takes the composition of two neural networks assuming they are compatible, i.e., given $\nu_1, \nu_2 \in \mathsf{NN}$, it must be the case that $\mathsf{I}(\nu)_1 = \mathsf{O}(\nu_2)$.

Usage

```
comp(phi_1, phi_2)

phi_1 %comp% phi_2
```

Arguments

phi_1	first neural network to be composed, goes on the left
phi_2	second neural network to be composed, goes on right

Value

The composed neural network. See also dep.

Our definition derive specifically from:

References

Definition 2.1.1. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360

Remark: We have two versions of this function, an infix version for close resemblance to mathematical notation and prefix version.

Examples

```
create_nn(c(5, 4, 6, 7)) |> comp(create_nn(c(4, 1, 5)))
```

Cpy *Cpy*

Description

The function that returns Cpy neural networks. These are neural networks defined as such

$$\mathsf{Aff}_{[\mathbb{I}_k \ \mathbb{I}_k \ \cdots \ \mathbb{I}_k]^T, 0_k}$$

Usage

```
Cpy(n, k)
```

Arguments

n	number of copies to make.
k	the size of the input vector.

Value

Returns an affine network that makes a concatenated vector that is n copies of the input vector of size k. See Aff and Sum.

References

Definition 2.9. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

create_block_diagonal *Function for creating a block diagonal given two matrices.*

Description

Function for creating a block diagonal given two matrices.

Usage

```
create_block_diagonal(matrix1, matrix2)
```

Arguments

matrix1	A matrix.
matrix2	A matrix

Value

A block diagonal matrix with matrix1 on top left and matrix2 on bottom right.

create_nn *create_nn*

Description

Function to create a list of lists for neural network layers

Usage

```
create_nn(layer_architecture)
```

Arguments

`layer_architecture`
a list specifying the width of each layer

Value

An ordered list of ordered pairs of (W, b). Where W is the matrix representing the weight matrix at that layer and b the bias vector. Entries on the matrix come from a standard normal distribution.

References

Definition 2.1 in Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Which in turn is a modified version of the one found in:

Definition 2.3. Grohs, P., Hornung, F., Jentzen, A. et al. Space-time error estimates for deep neural network approximations for differential equations. (2019). https://arxiv.org/abs/1908.03833.

Examples

```
create_nn(c(8, 7, 8))
create_nn(c(4,4))
```

`Csn` *Csn*

Description

The function that returns `Csn`.

Usage

```
Csn(n, q, eps)
```

Arguments

`n`	The number of Taylor iterations. Accuracy as well as computation time increases as n increases
`q`	a real number in $(2, \infty)$. Accuracy as well as computation time increases as q gets closer to 2 increases
`eps`	a real number in $(0, \infty)$. ccuracy as well as computation time increases as ε gets closer to 0 increases *Note:* In practice for most desktop uses $q < 2.05$ and $\varepsilon < 0.05$ tends to cause problems in "too long a vector", atleaast as tested on my computer.

Value

A neural network that approximates $\cos$ under instantiation with ReLU activation. See also `Sne`.

References

Definition 2.29 in Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Csn(2, 2.5, 0.5) # this may take some time

Csn(2, 2.5, 0.5) |> inst(ReLU, 1.50)
```

C_k — *C_k: The function that returns the C_k matrix*

Description

C_k: The function that returns the C_k matrix

Usage

```
C_k(k)
```

Arguments

k	Natural number, the precision with which to approximate squares within $[0, 1]$

Value

An intermediate matrix in a neural network that approximates the square of any real within $[0, 1]$ upon ReLU instantiation.

References

Definition 2.22. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
C_k(5)
```

dep — *dep*

Description

The function that returns the depth of a neural network. Denoted D.

Usage

```
dep(nu)
```

Arguments

nu	a neural network of the type generated by create_nn(). Very straightforwardly it is the length of the list where neural networks are defined as an odered list of lists.

Value

Integer representing the depth of the neural network.

References

Definition 1.3.1. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360.

Examples

```
create_nn(c(4, 5, 6, 2)) |> dep()
```

Etr *Etr*

Description

The function that returns the Etr networks.

Usage

```
Etr(n, h)
```

Arguments

n	number of trapezoids to make. Note this will result in a set of trapezoids. A natural number.
h	width of trapezoids. A positive real number. *Note:* Upon instantiation with any continuous function this neural network must be fed with $n+1$ real numbers representing the values of the function being approximated at the $n+1$ meshpoints which are the legs of the n trapezoids as stipulated in the input parameter n..

Value

An approximation for value of the integral of a function. Must be instantiated with a list of $n+1$ reals

References

Definition 2.33. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Etr(5, 0.1)
seq(0, pi, length.out = 1000) |> sin() -> samples
Etr(1000 - 1, pi / 1000) |> inst(ReLU, samples)

seq(0, 2, length.out = 1000)^2 -> samples
Etr(1000 - 1, 2 / 1000) |> inst(Tanh, samples)
```

---

generate_random_matrix   *Function to generate a random matrix with specified dimensions.*

Description

Function to generate a random matrix with specified dimensions.

Usage

```
generate_random_matrix(rows, cols)
```

Arguments

rows	number of rows.
cols	number of columns.

Value

a random matrix of dimension rows times columns with elements from a standard normal distribution

---

hid   *hid*

Description

The function that returns the number of hidden layers of a neural network. Denoted H

Usage

```
hid(nu)
```

Arguments

nu	a neural network of the type generated by create_nn() By definition $\mathsf{H}(\nu) = \mathsf{D}(\nu) - 1$

Value

Integer representing the number of hidden layers.

References

Definition 1.3.1. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360.

Examples

```
create_nn(c(4, 5, 6, 2)) |> hid()
```

`i` *i*

Description

The function that returns the $\beth$ network.

Usage

```
i(d)
```

Arguments

`d`	the size of the i network

Value

returns the i_d network

References

Definition 2.2.6. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360

Examples

```
i(5)
i(10)
```

`Id` *: Id*

Description

The function that returns the Id_1 networks.

Usage

```
Id(d = 1)
```

Arguments

`d`	the dimension of the Id network, by default it is 1.

Value

Returns the Id_1 network.

References

Definition 2.17. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Id()
Id(3)
```

`inn` *inn*

Description

The function that returns the input layer size of a neural network. Denoted I

Usage

```
inn(nu)
```

Arguments

`nu`	A neural network of the type generated by create_nn().

Value

An integer representing the input width of the neural network.

References

Definition 1.3.1. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360.

Examples

```
create_nn(c(4, 5, 6, 2)) |> inn()
```

`inst` *inst*

Description

The function that instantiates a neural network as created by create_nn().

Usage

```
inst(neural_network, activation_function, x)
```

Arguments

`neural_network` An ordered list of lists, of the type generated by create_nn() where each element in the list of lists is a pair (W, b) representing the weights and biases of that layer.

NOTE: We will call istantiation what Grohs et. al. call "realization".

`activation_function` A continuous function applied to the output of each layer. For now we only have ReLU, Sigmoid, and Tanh. Note, all proofs are only valid for ReLU activation.

`x` our input to the continuous function formed from activation. Our input will be an element in $\mathbb{R}^d$ for some appropriate d.

Value

The output of the continuous function that is the instantiation of the given neural network with the given activation function at the given x. Where x is of vector size equal to the input layer of the neural network.

References

Grohs, P., Hornung, F., Jentzen, A. et al. Space-time error estimates for deep neural network approximations for differential equations. (2019). https://arxiv.org/abs/1908.03833.

Definition 1.3.4. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360

Very precisely we will use the definition in:

Definition 2.3 in Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
create_nn(c(1, 3, 5, 6)) |> inst(ReLU, 5)
create_nn(c(3, 3, 5, 6)) |> inst(ReLU, c(4, 4, 4))
```

---

is_nn *is_nn*

Description

Function to create a list of lists for neural network layers

Usage

```
is_nn(nn)
```

Arguments

nn	A neural network. Neural networks are defined to be an ordered list of ordered pairs of (W, b). Where W is the matrix representing the weight matrix W at that layer and b the bias vector.

Value

TRUE or FALSE on whether nn is indeed a neural network as defined above.

We will use the definition of neural networks as found in:

References

Definition 2.1 in Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Which in turn is a modified version of the one found in:

Definition 2.3. Grohs, P., Hornung, F., Jentzen, A. et al. Space-time error estimates for deep neural network approximations for differential equations. (2019). https://arxiv.org/abs/1908.03833.

Examples

```
create_nn(c(5, 6, 7)) |> is_nn()
Sqr(2.1, 0.1) |> is_nn()
```

`lay` *lay*

Description

The function that returns the layer architecture of a neural network.

Usage

```
lay(nu)
```

Arguments

nu	A neural network of the type generated by create_nn(). Denoted L.

Value

A tuple representing the layer architecture of our neural network.

References

Definition 1.3.1. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360.

Examples

```
create_nn(c(4, 5, 6, 2)) |> lay()
```

`MC` *The MC neural network*

Description

This function implements the 1-D approximation scheme outlined in the References.

Note: Only 1-D interpolation is implemented.

Usage

```
MC(X, y, L)
```

Arguments

`X`	a list of samples from the functions domain.
`y`	the function applied componentwise to each point in the domain.
`L`	the Lipschitz constant for the function. Not necessarily global, but could be an absolute upper limit of slope, over the domain.

Value

A neural network that gives the maximum convolution approximation of a function whose outputs is y at n sample points given by each row of X, when instantiated with ReLU.

References

Lemma 4.2.9. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360.

Examples

```
seq(0, 3.1416, length.out = 200) -> X
sin(X) -> y
MC(X, y, 1) |> inst(ReLU, 0.25) # compare to sin(0.25)
```

Mxm *Mxm*

Description

The function that returns the Mxm neural networks.

Note: Because of certain quirks of R we will have split into five cases. We add an extra case for $d = 3$. Unlike the paper we will simply reverse engineer the appropriate d.

Usage

```
Mxm(d)
```

Arguments

`d`	The dimension of the input vector on instantiation.

Value

The neural network that will ouput the maximum of a vector of size d when activated with the ReLU function.

For a specific definition, see:

References

Lemma 4.2.4. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360

Examples

```
Mxm(1) |> inst(ReLU, -5)
Mxm(3) |> inst(ReLU, c(4, 5, 1))
Mxm(5) |> inst(ReLU, c(5, 3, -1, 6, 6))
```

---

nn_sum *nn_sum*

Description

A function that performs the neural network sum for two neural networks of the type generated by create_nn().

For a specific definition, see:

Usage

```
nn_sum(nu_1, nu_2)

nu_1 %nn_sum% nu_2
```

Arguments

nu_1	A neural network.
nu_2	A neural network.

Value

A neural network that is the neural network sum of ν_1 and ν_2 i.e. $\nu_1 \oplus \nu_2$.

Note: We have two versions, an infix version and a prefix version.

References

Proposition 2.25. Grohs, P., Hornung, F., Jentzen, A. et al. Space-time error estimates for deep neural network approximations for differential equations. (2019). https://arxiv.org/abs/1908.03833.

Examples

```
Prd(2.1, 0.1) |> nn_sum(Prd(2.1, 0.1))
```

`Nrm` *Nrm*

Description

A function that creates the `Nrm` neural networks.that take the 1- norm of a d-dimensional vector when instantiated with ReLU activation.

Usage

```
Nrm(d)
```

Arguments

`d`	the dimensions of the vector or list being normed.

Value

a neural network that takes the 1-norm of a vector of size d.under ReLU activation.

Note: This function is split into two cases much like the definition itself.

Note: If you choose to specify a d other that 0 you must instantiate with a vector or list of that length.

For a specific definition, see:

References

Lemma 4.2.1. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360

Examples

```
Nrm(2) |> inst(ReLU, c(5,6))
Nrm(5) |> inst(ReLU,c(0,-9,3,4,-11))
```

`out` *out*

Description

The function that returns the output layer size of a neural network. Denoted O.

Usage

```
out(nu)
```

Arguments

nu A neural network of the type generated by create_nn().

Value

An integer representing the output width of the neural network.

References

Definition 1.3.1. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360.

Examples

```
create_nn(c(4, 5, 6, 2)) |> out()
```

`param` *param*

Description

The function that returns the numbe of parameters of a neural network.

Usage

```
param(nu)
```

Arguments

nu A neural network of the type generated by create_nn(). Denoted P.

Value

An integer representing the parameter count of our neural network.

References

Definition 1.3.1. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360.

Examples

```
create_nn(c(4, 5, 6, 2)) |> param()
```

Phi *The Phi function*

Description

The Phi function

Usage

```
Phi(eps)
```

Arguments

eps	parameter for Phi in $(0, \infty)$

Value

neural network Phi that approximately squares a number between 0 and 1.

References

Definition 2.23. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Phi(0.5) |> view_nn()
Phi(0.1) |> view_nn()
```

Phi_k *The Phi_k function*

Description

The Phi_k function

Usage

```
Phi_k(k)
```

Arguments

k	an integer $k \in (2, \infty)$

Value

The Phi_k neural network

References

Definition 2.22. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Phi_k(4) |> view_nn()
Phi_k(5) |> view_nn()
```

Prd *Prd*

Description

A function that returns the Prd neural networks that approximates the product of two real numbers when given an appropriate q, ε, a real number x and instantiation with ReLU. activation.

Usage

```
Prd(q, eps)
```

Arguments

`q`	a real number in $(2, \infty)$. Accuracy as well as computation time increases as q gets closer to 2 increases
`eps`	a real number in $(0, \infty)$. ccuracy as well as computation time increases as ε gets closer to 0 increases

Value

A neural network that takes in x and y and approximately returns xy when instantiated with ReLU activation, and given a list c(x,y), the two numbers to be multiplied.

Note that this must be instantiated with a tuple c(x,y)

References

Proposition 3.5. Grohs, P., Hornung, F., Jentzen, A. et al. Space-time error estimates for deep neural network approximations for differential equations. (2019). https://arxiv.org/abs/1908.03833

Definition 2.25. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Prd(2.1, 0.1) |> inst(ReLU, c(4, 5)) # This may take some time, please only click once
```

`Pwr` *Pwr*

Description

A function that returns the Pwr neural networks.

Usage

```
Pwr(q, eps, exponent)
```

Arguments

`q`	a real number in $(2, \infty)$. Accuracy as well as computation time increases as q gets closer to 2 increases
`eps`	a real number in $(0, \infty)$. ccuracy as well as computation time increases as ε gets closer to 0 increases
`exponent`	The power to which we will raise. Computation time increases as exponent increases

Value

A neural network that approximates raising a number to exponent, when given appropriate q, ε and exponent when instantiated under ReLU activation at x.

Examples

```
Pwr(2.1, 0.1, 2) |> inst(ReLU, 3) # This may take some time, please only click once.
```

`ReLU` *: ReLU*

Description

The ReLU activation function

Usage

```
ReLU(x)
```

Arguments

`x`	A real number that is the input to our ReLU function.

Value

The output of the standard ReLU function, i.e. $\max\{0, x\}$. See also `Sigmoid`. and `Tanh`.

Examples

```
ReLU(5)
ReLU(-5)
```

`Sigmoid` *: Sigmoid*

Description

The Sigmoid activation function.

Usage

```
Sigmoid(x)
```

Arguments

x a real number that is the input to our Sigmoid function.

Value

The output of a standard Sigmoid function, i,e. $\frac{1}{1+\exp(-x)}$. See also Tanh.and ReLU.

Examples

```
Sigmoid(0)
Sigmoid(-1)
```

---

slm *slm*

Description

The function that returns the left scalar multiplication neural network

Usage

```
slm(a, nu)

a %slm% nu
```

Arguments

a	A real number.
nu	A neural network of the type generated by create_nn().

Value

Returns a neural network that is $a \triangleright \nu$. This instantiates as $a \cdot f(x)$ under continuous function activation. More specifically we define operation as:

Let $\lambda \in \mathbb{R}$. We will denote by $(\cdot) \triangleright (\cdot) : \mathbb{R} \times \mathsf{NN} \to \mathsf{NN}$ the function satisfying for all $\nu \in \mathsf{NN}$ and $\lambda \in \mathbb{R}$ that $\lambda \triangleright \nu = \mathsf{Aff}_{\lambda \mathbb{I}_{\mathsf{I}(\nu)},0} \bullet \nu$.

References

Definition 2.3.4. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360.

Note: We will have two versions of this operation, a prefix and an infix version.

Examples

```
5 |> slm(Prd(2.1, 0.1))
Prd(2.1, 0.1) |> srm(5)
```

Sne *Sne*

Description

Returns the Sne neural networks

Usage

```
Sne(n, q, eps)
```

Arguments

n	The number of Taylor iterations. Accuracy as well as computation time increases as n increases
q	a real number in $(2, \infty)$. Accuracy as well as computation time increases as q gets closer to 2 increases
eps	a real number in $(0, \infty)$. ccuracy as well as computation time increases as ε gets closer to 0 increases *Note:* In practice for most desktop uses $q < 2.05$ and $\varepsilon < 0.05$ tends to cause problems in "too long a vector", atleaast as tested on my computer.

Value

A neural network that approximates $\sin$ when given an appropriate n, q, ε and instantiated with ReLU activation and given value x.

References

Definition 2.30. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Sne(2, 2.3, 0.3) # this may take some time, click only once and wait

Sne(2, 2.3, 0.3) |> inst(ReLU, 1.57) # this may take some time, click only once and wait
```

`Sqr` *Sqr*

Description

A function that returns the Sqr neural networks.

Usage

```
Sqr(q, eps)
```

Arguments

q	a real number in $(2, \infty)$. Accuracy as well as computation time increases as q gets closer to 2 increases
eps	a real number in $(0, \infty)$. ccuracy as well as computation time increases as ε gets closer to 0 increases

Value

A neural network that approximates the square of a real number.when provided appropriate q, ε and upon instantiation with ReLU, and a real number x

References

Proposition 3.4. Grohs, P., Hornung, F., Jentzen, A. et al. Space-time error estimates for deep neural network approximations for differential equations. (2019). https://arxiv.org/abs/1908.03833

#' @references Definition 2.24. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Sqr(2.5, 0.1)
Sqr(2.5, 0.1) |> inst(ReLU, 4)
```

srm	*srm*

Description

The function that returns the right scalar multiplication neural network

Usage

```
srm(nu, a)

nu %srm% a
```

Arguments

nu	A neural network
a	A real number.

Value

Returns a neural network that is $\nu \triangleleft a$. This instantiates as $f(a \cdot x)$.under continuous function activation. More specifically we will define this operation as:

Let $\lambda \in \mathbb{R}$. We will denote by $(\cdot) \triangleleft (\cdot) : \mathsf{NN} \times \mathbb{R} \to \mathsf{NN}$ the function satisfying for all $\nu \in \mathsf{NN}$ and $\lambda \in \mathbb{R}$ that $\nu \triangleleft \lambda = \nu \bullet \mathsf{Aff}_{\lambda \mathbb{I}_{\mathsf{I}(\nu)},0}$.

References

Definition 2.3.4. Jentzen, A., Kuckuck, B., and von Wurstemberger, P. (2023). Mathematical introduction to deep learning: Methods, implementations, and theory. https://arxiv.org/abs/2310.20360.

Note: We will have two versions of this operation, a prefix and an infix version.

stk	*stk*

Description

A function that stacks neural networks.

Usage

```
stk(nu, mu)

nu %stk% mu
```

Arguments

nu	neural network.
mu	neural network.

Value

A stacked neural network of ν and μ, i.e. $\nu \boxminus \mu$

NOTE: This is different than the one given in Grohs, et. al. 2023. While we use padding to equalize neural networks being parallelized our padding is via the Tun network whereas Grohs et. al. uses repetitive composition of the i network. We use repetitive composition of the Id_1 network. See Id comp

NOTE: The terminology is also different from Grohs et. al. 2023. We call stacking what they call parallelization. This terminology change was inspired by the fact that parallelization implies commutativity but this operation is not quite commutative. It is commutative up to transposition of our input x under instantiation with a continuous activation function.

Also the word parallelization has a lot of baggage when it comes to artificial neural networks in that it often means many different CPUs working together.

Remark: We will use only one symbol for stacking equal and unequal depth neural networks, namely "stk". This is for usability but also that for all practical purposes only the general stacking of neural networks of different sizes is what is needed.

Remark: We have two versions, a prefix and an infix version.

This operation on neural networks, called "parallelization" is found in:

A stacked neural network of nu and mu.

References

Grohs, P., Hornung, F., Jentzen, A. et al. Space-time error estimates for deep neural network approximations for differential equations. (2023). https://arxiv.org/abs/1908.03833

And especially in:

' Definition 2.14 in Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
create_nn(c(4,5,6)) |> stk(create_nn(c(6,7)))
create_nn(c(9,1,67)) |> stk(create_nn(c(4,4,4,4,4)))
```

`Sum` *Sum*

Description

The function that returns Sum neural networks.

These are neural networks defined as such

$$\mathsf{Aff}_{[\mathbb{I}_k\ \mathbb{I}_k\ \cdots\ \mathbb{I}_k],0_k}$$

Usage

```
Sum(n, k)
```

Arguments

`n`	number of copies of a certain vector to be summed.
`k`	the size of the summation vector.

Value

An affine neural network that will take a vector of size $n \times k$ and return the summation vector that is of length k. See also `Aff` and `Cpy`.

References

Definition 2.10. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

`Tanh` *Tanh*

Description

The tanh activation function

Usage

```
Tanh(x)
```

Arguments

`x`	a real number

Value

the $tanh$ of x. See also Sigmoid and ReLU.

Examples

```
Tanh(0)
Tanh(0.1)
```

Tay *The Tay function*

Description

The Tay function

Usage

```
Tay(f, n, q, eps)
```

Arguments

f	the function to be Taylor approximated, for now "exp", "sin" and "cos". NOTE use the quotation marks when using this argument.
n	The number of Taylor iterations. Accuracy as well as computation time increases as n increases
q	a real number in $(2, \infty)$. Accuracy as well as computation time increases as q gets closer to 2 increases
eps	a real number in $(0, \infty)$. ccuracy as well as computation time increases as ε gets closer to 0 increases

Value

a neural network that approximates the function f. For now only sin, cos, and e^x are available.

Examples

```
Tay("sin", 2, 2.3, 0.3) |> inst(ReLU, 1.5) # May take some time, please only click once
Tay("cos", 2, 2.3, 0.3) |> inst(ReLU, 1) # May take some time, please only click once
Tay("exp", 4, 2.3, 0.3) |> inst(ReLU, 1.5) # May take some time, please only click once
```

Trp *Trp*

Description

The function that returns the Trp networks.

Usage

```
Trp(h)
```

Arguments

h the horizontal distance between two mesh points

Value

The Trp network that gives the area when activated with ReLU or any continuous function and two meshpoint values x_1 and x_2.

References

Definition 2.31. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Trp(0.1)
Trp(0.5) |> inst(ReLU, c(9, 7))
Trp(0.1) |> inst(Sigmoid, c(9, 8))
```

Tun *Tun: The function that returns tunneling neural networks*

Description

Tun: The function that returns tunneling neural networks

Usage

```
Tun(n, d = 1)
```

Arguments

n The depth of the tunnel network where $n \in \mathbb{N} \cap [1, \infty)$.

d The dimension of the tunneling network. By default it is assumed to be 1.

Value

A tunnel neural network of depth n. A tunneling neural network is defined as the neural network $\mathsf{Aff}_{1,0}$ for $n = 1$, the neural network Id_1 for $n = 1$ and the neural network $\bullet^{n-2}\mathsf{Id}_1$ for $n > 2$. For this to work we must provide an appropriate n and instantiate with ReLU at some real number x.

References

Definition 2.17. Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Tun(4)
Tun(4, 3) |> view_nn()

Tun(5)
Tun(5, 3)
```

---

view_nn *view_nn*

Description

Takes a neural network shown in vectorized form and explicitly displays it.

Usage

```
view_nn(nn)
```

Arguments

nn A neural network., i.e. a list of lists of W and b.

Value

A displayed version of the neural network. This may be required if the neural network is very deep.

Examples

```
c(5, 6, 7, 9) |>
  create_nn() |>
  view_nn()
Sqr(2.1, 0.1) |> view_nn()

Xpn(3, 2.1, 1.1) |> view_nn()
Pwr(2.1, 0.1, 3) |> view_nn()
```

Xpn *The Xpn function*

Description

The Xpn function

Usage

```
Xpn(n, q, eps)
```

Arguments

n	The number of Taylor iterations. Accuracy as well as computation time increases as n increases
q	a real number in $(2, \infty)$. Accuracy as well as computation time increases as q gets closer to 2 increases
eps	a real number in $(0, \infty)$. ccuracy as well as computation time increases as ε gets closer to 0 increases *Note:* In practice for most desktop uses $q < 2.05$ and $\varepsilon < 0.05$ tends to cause problems in "too long a vector", atleaast as tested on my computer.

Value

A neural network that approximates e^x for real x when given appropriate n, q, ε and instnatiated with ReLU activation at point$x$.

References

Definition 2.28 in Rafi S., Padgett, J.L., Nakarmi, U. (2024) Towards an Algebraic Framework For Approximating Functions Using Neural Network Polynomials https://arxiv.org/abs/2402.01058

Examples

```
Xpn(3, 2.25, 0.25) # this may take some time

Xpn(3, 2.2, 0.2) |> inst(ReLU, 1.5) # this may take some time
```

Index