# Package 'attention'

July 12, 2022

**Title** Self-Attention Algorithm

**Version** 0.2.0

**Description** Self-Attention algorithm helper functions and demonstration vignettes of increasing depth on how to construct the Self-Attention algorithm, this is based on Vaswani et al. (2017) <doi:10.48550/arXiv.1706.03762>, Dan Jurafsky and James H. Martin (2022, ISBN:978-0131873216) <https://web.stanford.edu/~jurafsky/slp3/> ``Speech and Language Processing (3rd ed.)'' and Alex Graves (2020) <https://www.youtube.com/watch?v=AIiwuClvH6k> ``Attention and Memory in Deep Learning''.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.2.0

**Suggests** covr, knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Bastiaan Quast [aut, cre] (<https://orcid.org/0000-0002-2951-3577>)

**Maintainer** Bastiaan Quast <bquast@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-07-12 14:20:15 UTC

## R topics documented:

attention *Self-Attention Algorithm*

## Description

Implements functions for the Self-Attention algorithm, and demonstrates with vignettes of increasing depth how to construct the Self-Attention algorithm..

## Author(s)

Bastiaan Quast <bquast@gmail.com>

## References

https://qua.st/attention

## See Also

https://qua.st/attention

## Examples

```
# encoder representations of four different words
word_1 = matrix(c(1,0,0), nrow=1)
word_2 = matrix(c(0,1,0), nrow=1)
word_3 = matrix(c(1,1,0), nrow=1)
word_4 = matrix(c(0,0,1), nrow=1)

# stacking the word embeddings into a single array
words = rbind(word_1,
              word_2,
              word_3,
              word_4)

# generating the weight matrices
set.seed(0)
W_Q = matrix(floor(runif(9, min=0, max=3)),nrow=3,ncol=3)
W_K = matrix(floor(runif(9, min=0, max=3)),nrow=3,ncol=3)
W_V = matrix(floor(runif(9, min=0, max=3)),nrow=3,ncol=3)

# generating the queries, keys and values
Q = words %*% W_Q
K = words %*% W_K
V = words %*% W_V

# scoring the query vectors against all key vectors
scores = Q %*% t(K)

# calculate the max for each row of the scores matrix
maxs = as.matrix(apply(scores, MARGIN=1, FUN=max))
```

```
# initialize weights matrix
weights = matrix(0, nrow=4, ncol=4)

# computing the weights by a softmax operation
for (i in 1:dim(scores)[1]) {
 weights[i,] = exp((scores[i,]-maxs[i,]) /
               ncol(K) ^ 0.5)/sum(exp((scores[i,]-maxs[i,]) / ncol(K) ^ 0.5))
}

# computing the attention by a weighted sum of the value vectors
attention = weights %*% V

print(attention)
```

---

ComputeWeights *SoftMax sigmoid function*

---

## Description

SoftMax sigmoid function

## Usage

```
ComputeWeights(scores)
```

## Arguments

| | |
|---|---|
| scores | input value (numeric) |

## Value

output value (numeric)

## Examples

```
# Set up a scores matrix
scores <- matrix(c( 6,  4, 10,  5,
                    4,  6, 10,  6,
                   10, 10, 20, 11,
                    3,  1,  4,  2),
                 nrow  = 4,
                 ncol  = 4,
                 byrow = TRUE)

# Compute the weights based on the scores matrix
ComputeWeights(scores)

# this outputs
#            [,1]       [,2]      [,3]       [,4]
```

```
# [1,] 0.10679806 0.03928881 0.7891368 0.06477630
# [2,] 0.03770440 0.10249120 0.7573132 0.10249120
# [3,] 0.00657627 0.00657627 0.9760050 0.01084244
# [4,] 0.27600434 0.10153632 0.4550542 0.16740510
```

---

RowMax *Maximum of Matrix Rows*

---

## Description

Maximum of Matrix Rows

## Usage

```
RowMax(x)
```

## Arguments

| | |
|---|---|
| x | input value (numeric) |

## Value

output value (numeric)

## Examples

```
# generate a matrix of integers (also works for floats)
set.seed(0)
M = matrix(floor(runif(9, min=0, max=3)),
           nrow=3,
           ncol=3)
print(M)

# this outputs
#      [,1] [,2] [,3]
# [1,]    2    1    2
# [2,]    0    2    2
# [3,]    1    0    1

# apply RowMax() to the matrix M, reformat output as matrix again
# to keep the maxs on their corresponding rows
RowMax(M)

# this outputs
#      [,1]
# [1,]    2
# [2,]    2
# [3,]    1
```

SoftMax *SoftMax sigmoid function*

## Description

SoftMax sigmoid function

## Usage

```
SoftMax(x)
```

## Arguments

| | |
|---|---|
| x | input value (numeric) |

## Value

output value (numeric)

## Examples

```
# create a vector of integers (as works for non-integers)
set.seed(0)
V = c(floor(runif(9, min=-3, max=3)))
print(V)

# this outputs
# [1]  2 -2 -1  0  2 -2  2  2  0

# apply the SoftMax() function to V
sV <- SoftMax(V)
print(sV)

# this outputs
# [1] 0.229511038 0.004203641 0.011426682 0.031060941
# 0.229511038 0.004203641 0.229511038 0.229511038 0.031060941
```

# Index