Package: aimat (via r-universe)

June 4, 2026

Type Package

Title Functions to support aimat.dk

Version 0.1.1

Description Functions to support the Danish language site aimat.dk containing teaching material related to the math behind AI.

Imports Matrix

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 4.2)

Suggests knitr, visNetwork, dplyr, plotly, shiny

Repository https://aalborg-intelligence.r-universe.dev

Date/Publication 2026-06-04 22:00:30 UTC

RemoteUrl https://github.com/aalborg-intelligence/aimat

RemoteRef HEAD

RemoteSha ef1b8448008642e770d3a3700e6e8f9ccfeaf2ee

Contents

`aimatApp` *Run an aimat Shiny app*

Description

This function launches a Shiny app for training neural networks using the ‘aimat‘ package.

Usage

```
aimatApp(appname)
```

Arguments

`appname`	A character string specifying the name of the app to run. The app should be located in the ‘aimat‘ package’s system file directory.

Examples

```
if (interactive() & requireNamespace("shiny", quietly = TRUE)) {
  aimatApp("app_full")
}
```

`bog` *Words in an artificial childrens book (in Danish)*

Description

Words in an artificial childrens book (in Danish)

Usage

```
bog
```

Format

‘bog‘ A vector of words

fibonacci_sphere *Fibonacci Sphere*

Description

Fibonacci Sphere

Usage

```
fibonacci_sphere(samples = 1000)
```

Arguments

samples	Number of points to generate on the sphere.

Value

A matrix of points on the sphere, where each row is a point in 3D space.

Examples

```
points <- fibonacci_sphere()
if(requireNamespace("plotly", quietly = TRUE)){
  plotly::plot_ly(x = points[, 1], y = points[, 2], z = points[, 3],
    type = "scatter3d", mode = "markers", marker = list(size = 5))
}
```

nn_fun *General neural network with at most two hidden layers*

Description

General neural network with at most two hidden layers

Usage

```
nn_fun(
  formula,
  data,
  weights = NA,
  n_hidden = c(1, 1),
  activation = "Sigmoid",
  eta = 0.01,
  iter = 1000,
  scale = TRUE,
```

```
  lossfun = "squared",
  type = "klassifikation",
  trace = FALSE
)
```

Arguments

formula	response ~ predictors
data	data frame
weights	Starting weights
n_hidden	integer vector of length 2 with number of neurons in the hidden layers (can be zero)
activation	one of "Sigmoid", "Tangenshyperbolsk", "ReLu", "Softsign", "Identitet"
eta	learning rate
iter	number of iterations
scale	logical to scale the input data
lossfun	one of "squared", "cross-entropy"
type	one of "regression", "klassifikation"
trace	logical to save each iteration

Value

Fitted neural network model as an object of class "nn"

Examples

```
ir <- iris
ir[,1:4] <- scale(ir[,1:4])
fit_ir <- nn_fun(Species ~ ., ir, n_hidden = c(3,5), eta = 0.01, iter = 1000,
  lossfun = "cross-entropy", activation = "Sigmoid", type = "klassifikation")
```

---

nn_fun_cv *Cross-validation for neural network*

Description

Cross-validation for neural network

Usage

```
nn_fun_cv(formula, data, ..., k = 5)
```

Arguments

formula	response ~ predictors
data	data frame
...	Additional arguments passed to ‘nn_fun‘
k	Number of folds for cross-validation

Value

Mean accuracy across folds

Examples

```
cv <- nn_fun_cv(Species ~ ., iris, n_hidden = c(3,5), eta = 0.01, iter = 1000,
lossfun = "cross-entropy")
```

---

nn_viz *Visualize a neural network*

Description

Visualize a neural network

Usage

```
nn_viz(nn)
```

Arguments

nn	FIXME

Examples

```
ir <- iris
ir[,1:4] <- scale(ir[,1:4])
fit_ir <- nn_fun(Species ~ ., ir, n_hidden = c(3,5), eta = 0.01, iter = 1000,
  lossfun = "cross-entropy", activation = "Sigmoid", type = "klassifikation")
if(requireNamespace("visNetwork", quietly = TRUE)){
  nn_viz(fit_ir)
}
```

nwp_make_word_data *Make next word prediction data*

Description

Make next word prediction data

Usage

```
nwp_make_word_data(corpus, context_size, w2v, return = "both")
```

Arguments

corpus	Vector of words.
context_size	Integer, size of the context window (using this many words to predict the next).
w2v	Word2Vec model object. This is a list with three elements: ‘W1‘, ‘W2‘ and ‘vocab‘. ‘W1‘ and ‘W2‘ are matrices of word vectors, and ‘vocab‘ is a vector of words.
return	Character, one of "both", "words" or "numeric". If "both", returns a list with two elements: ‘words‘ and ‘numeric‘. If "words", returns only the ‘words‘ element. If "numeric", returns only the ‘numeric‘ element. Default is "both".

Value

A list containing data to train next word prediction. The list contains one or both of the following elements: - words: a data.frame with ‘context_size‘+1 columns of words where the last column is the target word. - numeric: a numeric matrix

Examples

```
bog_w2v <- w2v(bog, win = 1, hidden_dim = 3, epochs = 5, learning_rate = 0.01, verbose = 0)
bog_data <- nwp_make_word_data(corpus = bog, context_size = 2, w2v = bog_w2v)
```

plot.w2v *Plot word2vec model*

Description

Plot word2vec model

Usage

```
## S3 method for class 'w2v'
plot(x, ..., which = 1)
```

Arguments

x	Word2Vec model object (list with W1, W2 and vocab)
...	Additional arguments (not used)
which	Integer, 1 or 2. If 1, plot W1 (input layer), if 2, plot W2 (output layer)

Examples

```
w <- w2v(bog, hidden_dim = 2, epochs = 10, learning_rate = 0.01, verbose = 1)
plot(w, which = 1)
```

---

predict.nn *Predict method for neural network*

Description

Predict method for neural network

Usage

```
## S3 method for class 'nn'
predict(object, newdata, type = "response", ...)
```

Arguments

object	Fitted neural network model
newdata	Data frame with new data
type	One of "response", "class"
...	Additional arguments, currently not used

Value

Predicted values

Examples

```
nn_iris <- nn_fun(Species ~ ., iris, n_hidden = c(3,5), eta = 0.01, iter = 1000,
  lossfun = "cross-entropy", activation = "Sigmoid",
  type = "klassifikation", scale = TRUE)
predict(nn_iris, iris, type = "class")

nn_trees <- nn_fun(Volume ~ ., trees, n_hidden = c(3,5), eta = 0.01, iter = 1000,
  lossfun = "squared", activation = "Sigmoid",
  type = "regression", scale = TRUE)
predict(nn_trees, trees, type = "response")
```

w2v *Word2vec Neural Network*

Description

Word2vec Neural Network

Usage

```
w2v(
  corpus,
  win = 1,
  hidden_dim = 3,
  epochs = 100,
  learning_rate = 0.01,
  verbose = 10,
  weights = NULL
)
```

Arguments

corpus	Vector of words
win	Integer, size of the context window
hidden_dim	Dimension of the hidden layer
epochs	Number of training epochs
learning_rate	Learning rate
verbose	Integer, verbosity level (0 = no output, >0 = output every ‘verbose‘ epochs)
weights	FIXME

Examples

```
w <- w2v(bog, hidden_dim = 2, epochs = 10, learning_rate = 0.01, verbose = 1)
```

w2v_build_data *Build word2vec data from a corpus represented as a vector of words*

Description

Build word2vec data from a corpus represented as a vector of words

Usage

```
w2v_build_data(corpus, win = 1)
```

Arguments

corpus	vector of words
win	integer, size of the context window

Value

A list containing the following elements: - ‘X‘: Sparse matrix of one-hot encoded input vectors - ‘Y‘: Sparse matrix of one-hot encoded output vectors - ‘vocab‘: Vector of unique words in the corpus - ‘pairs‘: Data frame of word pairs in context

Examples

```
w2v_data <- w2v_build_data(bog, win = 1)
```

---

w2v_nn *Word2vec Neural Network*

Description

Word2vec Neural Network

Usage

```
w2v_nn(X, Y, hidden_dim, epochs, learning_rate, weights = NULL, verbose = 10)
```

Arguments

X	Matrix of one-hot encoded input vectors
Y	Matrix of one-hot encoded output vectors
hidden_dim	Dimension of the hidden layer
epochs	Number of training epochs
learning_rate	Learning rate
weights	Initial weights for the network (random if NULL)
verbose	FIXME

Value

Named list contaning trained weights for the network

Examples

```
# Example usage
library(Matrix)
n <- 10  # Dimension of input and output
m <- 3   # Dimension of the first hidden layer

# Create input matrix with several one-hot vectors
X <- Matrix(0, nrow = 5, ncol = n, sparse = TRUE)
for (i in 1:5) {
  X[i, sample(1:n, 1)] <- 1
}

# Create output matrix with corresponding one-hot vectors
Y <- Matrix(0, nrow = 5, ncol = n, sparse = TRUE)
for (i in 1:5) {
  Y[i, sample(1:n, 1)] <- 1
}

# Train the network
trained_weights <- w2v_nn(X, Y, m, epochs = 100, learning_rate = 0.1, verbose = 10)
```

Index