

Package ‘simpleMLP’

October 14, 2022

Title Simple Multilayer Perceptron Neural Network

Version 1.0.0

Description Create and train a multilayer perceptron, a type of feedforward, fully connected neural network. Features 2 ReLU hidden layers. Learn more about about the activation functions and backpropagation used by this network in Goodfellow et al. (2016, ISBN: 9780262035613) ``Deep Learning".

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

Imports ggplot2, readr, stats

Depends R (>= 2.10)

NeedsCompilation no

Author Cullen Pu [aut, cre]

Maintainer Cullen Pu <cullenpu@gmail.com>

Repository CRAN

Date/Publication 2021-03-28 06:20:02 UTC

R topics documented:

| | |
|----------------------------|---|
| backprop | 2 |
| evaluate | 2 |
| forwardprop | 3 |
| init_nn | 3 |
| load_mnist | 4 |
| one_hot_encoding | 4 |
| plot_accuracy | 5 |
| train_nn | 5 |
| update | 6 |

| | |
|--------------|----------|
| Index | 8 |
|--------------|----------|

| | |
|----------|------------------------|
| backprop | <i>Backpropagation</i> |
|----------|------------------------|

Description

Runs a backwards pass through the network.

Usage

```
backprop(model, error, forward_pass)
```

Arguments

| | |
|--------------|---|
| model | list of all the weights and biases |
| error | gradients to the output of the network |
| forward_pass | intermediate values from the forward pass |

Value

list of derivatives after the backwards pass

| | |
|----------|-----------------------|
| evaluate | <i>Evaluate Model</i> |
|----------|-----------------------|

Description

Evaluates the performance of a model on a given dataset.

Usage

```
evaluate(inputs, target, model)
```

Arguments

| | |
|--------|--|
| inputs | set of inputs to the model |
| target | set of targets in one-hot encoded form |
| model | list of weights and biases |

Value

accuracy of the model

Examples

```
## Not run:  
evaluate(train_data, train_target, mlp_model)  
  
## End(Not run)
```

| | |
|-------------|----------------------------|
| forwardprop | <i>Forward propagation</i> |
|-------------|----------------------------|

Description

Runs a forward pass through the network.

Usage

```
forwardprop(model, x)
```

Arguments

| | |
|-------|------------------------------------|
| model | list of all the weights and biases |
| x | input to the network |

Value

list of all intermediate values

| | |
|---------|---------------------------|
| init_nn | <i>Initialize network</i> |
|---------|---------------------------|

Description

Initialize 3 layer fully connected neural network, also known as multilayer perceptron, setting biases to 0 and using the Xavier initialization method for weights.

Usage

```
init_nn(num_inputs, num_hidden_1, num_hidden_2, num_outputs)
```

Arguments

| | |
|--------------|----------------------------------|
| num_inputs | dimension of inputs |
| num_hidden_1 | dimension of first hidden layer |
| num_hidden_2 | dimension of second hidden layer |
| num_outputs | dimension of output |

Value

list containing weight and bias matrices in each layer of the network

Examples

```
mlp_model <- init_nn(784, 100, 50, 10)
```

| | |
|------------|---------------------------|
| load_mnist | <i>Load Training Data</i> |
|------------|---------------------------|

Description

Loads MNIST training, validation, and test data and generates one hot encodings for the targets. The test set proportion is not specified and is instead the remainder from the test and validation proportions.

Usage

```
load_mnist(train_prop = 0.8, validate_prop = 0.1)
```

Arguments

train_prop proportion of the data used for the training set
validate_prop proportion of the data used for the validation set

Value

list of training and validation data and targets

Examples

```
mnist <- load_mnist(0.8, 0.1)  
train_data <- mnist[1]  
train_target <- mnist[2]  
validate_data <- mnist[3]  
validate_target <- mnist[4]  
test_data <- mnist[5]  
test_target <- mnist[6]
```

| | |
|------------------|-------------------------|
| one_hot_encoding | <i>One Hot Encoding</i> |
|------------------|-------------------------|

Description

Creates a one hot encoding matrix with the specified number of categories for the targets. Target must be the first column of the data_raw input.

Usage

```
one_hot_encoding(data_raw, ncat = 10)
```

Arguments

data_raw data input to create encoding; target must be first column
ncat number of categories to use for the encoding

Value

targets in a one hot encoding matrix

| | |
|---------------|----------------------|
| plot_accuracy | <i>Plot Accuracy</i> |
|---------------|----------------------|

Description

Plot the training and validation accuracy.

Usage

```
plot_accuracy(accuracy_train, accuracy_validate)
```

Arguments

accuracy_train list of training accuracy
accuracy_validate list of validation accuracy

| | |
|----------|----------------------|
| train_nn | <i>Train Network</i> |
|----------|----------------------|

Description

Train the network with specified hyperparameters and return the trained model.

Usage

```
train_nn(  

  train_data,  

  train_target,  

  validate_data,  

  validate_target,  

  model,  

  alpha,  

  epochs,  

  batch_size = nrow(train_data),  

  plot_acc = TRUE  

)
```

Arguments

| | |
|-----------------|---|
| train_data | set of training data |
| train_target | set of training data targets in one-hot encoded form |
| validate_data | set of validation data targets in one-hot encoded form |
| validate_target | set of targets in |
| model | list of weights and biases |
| alpha | learning rate |
| epochs | number of epochs |
| batch_size | mini-batch size |
| plot_acc | whether or not to plot training and validation accuracy |

Value

list of weights and biases after training

Examples

```
## Not run:
mlp_model <- init_nn(784, 100, 50, 10)
mnist <- load_mnist()
train_data <- mnist[1]
train_target <- mnist[2]
validate_data <- mnist[3]
validate_target <- mnist[4]
mlp_model <- train_nn(train_data, train_target, validate_data,
validate_target, mlp_model, 0.01, 1, 64)

## End(Not run)
```

update

Update Model

Description

Updates the model using derivatives from a backward pass.

Usage

```
update(model, back_pass, alpha)
```

Arguments

| | |
|-----------|---|
| model | list of all the weights and biases |
| back_pass | derivatives from a backwards pass through the network |
| alpha | learning rate |

update

7

Value

updated list of the weights and biases

Index

backprop, [2](#)

evaluate, [2](#)

forwardprop, [3](#)

init_nn, [3](#)

load_mnist, [4](#)

one_hot_encoding, [4](#)

plot_accuracy, [5](#)

train_nn, [5](#)

update, [6](#)