# Package 'deepNN'

March 5, 2020

**Title** Deep Learning

**Version** 1.0

**Description** Implementation of some Deep Learning methods. Includes multilayer perceptron, different activation functions, regularisation strategies, stochastic gradient descent and dropout. Thanks go to the following references for helping to inspire and develop the package: Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach (2016, ISBN:978-0262035613) Deep Learning. Terrence J. Sejnowski (2018, ISBN:978-0262038034) The Deep Learning Revolution. Grant Sanderson (3brown1blue) <https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi> Neural Networks YouTube playlist. Michael A. Nielsen <http://neuralnetworksanddeeplearning.com/> Neural Networks and Deep Learning.

**Depends** R (>= 3.2.1)

**Imports** stats, graphics, utils, Matrix, methods

**License** GPL-3

**RoxygenNote** 7.0.2

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Benjamin Taylor [aut, cre]

**Maintainer** Benjamin Taylor <benjamin.taylor.software@gmail.com>

**Repository** CRAN

**Date/Publication** 2020-03-05 12:00:05 UTC

## R topics documented:

---

`deepNN-package` *deepNN*

---

## Description

Teaching resources (yet to be added) and implementation of some Deep Learning methods. Includes multilayer perceptron, different activation functions, regularisation strategies, stochastic gradient descent and dropout.

## Usage

```
deepNN
```

### Format

An object of class `logical` of length 1.

### Details

| | |
|---|---|
| Package: | deepNN |
| Version: | 0.1 |
| Date: | 2019-01-11 |
| License: | GPL-3 |

sectionDependencies The package `deepNN` depends upon some other important contributions to CRAN in order to operate; their uses here are indicated:

stats, graphics.

sectionCitation deepNN: Deep Learning. Benjamin M. Taylor

references Thanks go to the following references for helping to inspire and develop the package: Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach (2016, ISBN:978-0262035613) Deep Learning. Terrence J. Sejnowski (2018, ISBN:978-0262038034) The Deep Learning Revolution. Grant Sanderson (3brown1blue) <https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi> Neural Networks YouTube playlist. Michael A. Nielsen <http://neuralnetworksanddeeplearning.com/> Neural Networks and Deep Learning


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


### Author(s)

Benjamin Taylor, Department of Medicine, Lancaster University

---

## `addGrad` *addGrad function*

---

### Description

A function to add two gradients together, gradients expressed as nested lists.

### Usage

```
addGrad(x, y)
```

### Arguments

| | |
|---|---|
| x | a gradient list object, as used in network training via backpropagation |
| y | a gradient list object, as used in network training via backpropagation |

### Value

another gradient object

### References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


### See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

---

## addList *addList function*

---

### Description

A function to add two lists together

### Usage

```
addList(x, y)
```

### Arguments

| | |
|---|---|
| x | a list |
| y | a list |

### Value

a list, the elements of which are the sums of the elements of the arguments x and y.

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

---

`backpropagation_MLP` *backpropagation_MLP function*

---

## Description

A function to perform backpropagation for a multilayer perceptron.

## Usage

```
backpropagation_MLP(MLPNet, loss, truth)
```

## Arguments

| | |
|---|---|
| `MLPNet` | output from the function MLP_net, as applied to some data with given parameters |
| `loss` | the loss function, see ?Qloss and ?multinomial |
| `truth` | the truth, a list of vectors to compare with output from the feed-forward network |

## Value

a list object containing the cost and the gradient with respect to each of the model parameters

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/

## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

---

`backprop_evaluate` *backprop_evaluate function*

---

## Description

A function used by the train function in order to conduct backpropagation.

## Usage

```
backprop_evaluate(parameters, dat, truth, net, loss, batchsize, dropout)
```

## Arguments

| | |
|---|---|
| `parameters` | network weights and bias parameters as a vector |
| `dat` | the input data, a list of vectors |
| `truth` | the truth, a list of vectors to compare with output from the feed-forward network |
| `net` | an object of class network, see ?network |
| `loss` | the loss function, see ?Qloss and ?multinomial |
| `batchsize` | optional batchsize argument for use with stochastic gradient descent |
| `dropout` | optional list of dropout probabilities ?dropoutProbs |

## Value

the derivative of the cost function with respect to each of the parameters

## References

1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/

## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

`bias2list` *bias2list function*

## Description

A function to convert a vector of biases into a ragged array (coded here a list of vectors)

## Usage

```
bias2list(bias, dims)
```

## Arguments

| | |
|---|---|
| `bias` | a vector of biases |
| `dims` | the dimensions of the network as stored from a call to the function network, see ?network |

## Value

a list object with appropriate structures for compatibility with the functions network, train, MLP_net and backpropagation_MLP

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

`biasInit` *biasInit function*

## Description

A function to inialise memory space for bias parameters. Now redundant.

## Usage

```
biasInit(dims)
```

## Arguments

| | |
|---|---|
| `dims` | the dimensions of the network as stored from a call to the function network, see ?network |

## Value

memory space for biases

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

---

`download_mnist` *download_mnist function*

## Description

A function to download mnist data in .RData format. File includes objects train_set, truth, test_set and test_truth

## Usage

```
download_mnist(fn)
```

### Arguments

| | |
|---|---|
| fn | the name of the file to save as |

### Value

a list, the elements of which are the sums of the elements of the arguments x and y.

### References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/
5. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998
6. http://yann.lecun.com/exdb/mnist/


### See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP

### Examples

```
download_mnist("mnist.RData")
```

---

## dropoutProbs *dropoutProbs function*

---

### Description

A function to specify dropout for a neural network.

### Usage

```
dropoutProbs(input = 1, hidden = 1)
```

### Arguments

| | |
|---|---|
| input | inclusion rate for input parameters |
| hidden | inclusion rate for hidden parameters |

## Value

returns these probabilities in an appropriate format for interaction with the network and train functions, see ?network and ?train

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

## Examples

```
netwts <- train( dat=d,
                 truth=truth,
                 net=net,
                 eps=0.01,
                 tol=0.95,           # run for 100 iterations
                 batchsize=10,       # note this is not enough
                 loss=multinomial(), # for convergence
                 dropout=dropoutProbs(input=0.8,hidden=0.5))
```

---

`gradInit` *gradInit function*

---

## Description

A function to initialise memory for the gradient.

## Usage

```
gradInit(dim)
```

### Arguments

| | |
|---|---|
| dim | the dimensions of the network as stored from a call to the function network, see ?network |

### Value

memory space and structure for the gradient, initialised as zeros

### References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


### See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

---

## hyptan *hyptan function*

---

### Description

A function to evaluate the hyperbolic tanget activation function, the derivative and cost derivative to be used in defining a neural network.

### Usage

```
hyptan()
```

### Value

a list of functions used to compute the activation function, the derivative and cost derivative.

### References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/

## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, ReLU, smoothReLU, ident, softmax

## Examples

```
# Example in context

net <- network( dims = c(100,50,20,2),
                activ=list(hyptan(),ReLU(),softmax()))
```

---

`ident` *ident function*

---

## Description

A function to evaluate the identity (linear) activation function, the derivative and cost derivative to be used in defining a neural network.

## Usage

```
ident()
```

## Value

a list of functions used to compute the activation function, the derivative and cost derivative.

## References

1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/

## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, softmax

## Examples

```
# Example in context

net <- network( dims = c(100,50,20,2),
                activ=list(ident(),ReLU(),softmax()))
```

---

`L1_regularisation` *L1_regularisation function*

---

## Description

A function to return the L1 regularisation strategy for a network object.

## Usage

```
L1_regularisation(alpha)
```

## Arguments

| | |
|---|---|
| `alpha` | parameter to weight the relative contribution of the regulariser |

## Value

list containing functions to evaluate the cost modifier and grandient modifier

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, L2_regularisation, no_regularisation

## Examples

```
# Example in context: NOTE the value of 1 used here is arbitrary,
# to get this to work well, you'll have to experiment.

net <- network( dims = c(784,16,16,10),
                regulariser = L1_regularisation(1),
                activ=list(ReLU(),logistic(),softmax()))
```

L2_regularisation *L2_regularisation function*

## Description

A function to return the L2 regularisation strategy for a network object.

## Usage

```
L2_regularisation(alpha)
```

## Arguments

| | |
|---|---|
| alpha | parameter to weight the relative contribution of the regulariser |

## Value

list containing functions to evaluate the cost modifier and grandient modifier

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, L1_regularisation, no_regularisation

## Examples

```
# Example in context: NOTE the value of 1 used here is arbitrary,
# to get this to work well, you'll have to experiment.

net <- network( dims = c(784,16,16,10),
                regulariser = L2_regularisation(1),
                activ=list(ReLU(),logistic(),softmax()))
```

`logistic` *logistic function*

## Description

A function to evaluate the logistic activation function, the derivative and cost derivative to be used in defining a neural network.

## Usage

```
logistic()
```

## Value

a list of functions used to compute the activation function, the derivative and cost derivative.

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, ReLU, smoothReLU, ident, softmax

## Examples

```
# Example in context

net <- network( dims = c(100,50,20,2),
                activ=list(logistic(),ReLU(),softmax()))
```

## memInit *memInit function*

### Description

A function to initialise memory space. Likely this will become deprecated in future versions.

### Usage

```
memInit(dim)
```

### Arguments

| | |
|---|---|
| dim | the dimensions of the network as stored from a call to the function network, see ?network |

### Value

memory space, only really of internal use

### References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


### See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

## MLP_net *MLP_net function*

### Description

A function to define a multilayer perceptron and compute quantities for backpropagation, if needed.

### Usage

```
MLP_net(input, weights, bias, dims, nlayers, activ, back = TRUE, regulariser)
```

### Arguments

| | |
|---|---|
| input | input data, a list of vectors (i.e. ragged array) |
| weights | a list object containing weights for the forward pass, see ?weights2list |
| bias | a list object containing biases for the forward pass, see ?bias2list |
| dims | the dimensions of the network as stored from a call to the function network, see ?network |
| nlayers | number of layers as stored from a call to the function network, see ?network |
| activ | list of activation functions as stored from a call to the function network, see ?network |
| back | logical, whether to compute quantities for backpropagation (set to FALSE for feed-forward use only) |
| regulariser | type of regularisation strategy to, see ?train, ?no_regularisation ?L1_regularisation, ?L2_regularisation |

### Value

a list object containing the evaluated forward pass and also, if selected, quantities for backpropagation.

### References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


### See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

---

## multinomial *multinomial function*

---

### Description

A function to evaluate the multinomial loss function and the derivative of this function to be used when training a neural network.

### Usage

```
multinomial()
```

## Value

a list object with elements that are functions, evaluating the loss and the derivative

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, Qloss, no_regularisation, L1_regularisation, L2_regularisation

## Examples

```
netwts <- train(dat=train_set,
                truth=truth,
                net=net,
                eps=0.001,
                tol=0.95,
                loss=multinomial(),
                batchsize=100)
```

---

`nbiaspar` *nbiaspar function*

---

## Description

A function to calculate the number of bias parameters in a neural network, see ?network

## Usage

```
nbiaspar(net)
```

## Arguments

| | |
|---|---|
| `net` | an object of class network, see ?network |

## Value

an integer, the number of bias parameters in a neural network

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

## Examples

```
net <- network( dims = c(5,10,2),
                activ=list(ReLU(),softmax()))
nbiaspar(net)
```

---

`network` *network function*

---

## Description

A function to set up a neural network structure.

## Usage

```
network(dims, activ = logistic(), regulariser = NULL)
```

## Arguments

| | |
|---|---|
| `dims` | a vector giving the dimensions of the network. The first and last elements are respectively the input and output lengths and the intermediate elements are the dimensions of the hidden layers |
| `activ` | either a single function or a list of activation functions, one each for the hidden layers and one for the output layer. See for example ?ReLU, ?softmax etc. |
| `regulariser` | optional regularisation strategy, see for example ?no_regularisation (the default) ?L1_regularisation, ?L2_regularisation |

## Value

a list object with all information to train the network

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

## Examples

```
net <- network( dims = c(5,10,2),
                activ=list(ReLU(),softmax()))

net <- network( dims = c(100,50,50,20),
                activ=list(ReLU(),ReLU(),softmax()),
                regulariser=L1_regularisation())
```

---

`nnetpar` *nnetpar function*

---

## Description

A function to calculate the number of weight parameters in a neural network, see ?network

## Usage

```
nnetpar(net)
```

## Arguments

| | |
|---|---|
| `net` | an object of class network, see ?network |

## Value

an integer, the number of weight parameters in a neural network

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

## Examples

```
net <- network( dims = c(5,10,2),
                activ=list(ReLU(),softmax()))
nnetpar(net)
```

---

`NNgrad_test` *NNgrad_test function*

---

## Description

A function to test gradient evaluation of a neural network by comparing it with central finite differencing.

## Usage

```
NNgrad_test(net, loss = Qloss(), eps = 1e-05)
```

## Arguments

| | |
|---|---|
| `net` | an object of class network, see ?network |
| `loss` | a loss function to compute, see ?Qloss, ?multinomial |
| `eps` | small value used in the computation of the finite differencing. Default value is 0.00001 |

## Value

the exact (computed via backpropagation) and approximate (via central finite differencing) gradients and also a plot of one against the other.

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

## Examples

```
net <- network( dims = c(5,10,2),
                activ=list(ReLU(),softmax()))
NNgrad_test(net)
```

---

`NNpredict` *NNpredict function*

---

## Description

A function to produce predictions from a trained network

## Usage

```
NNpredict(
  net,
  param,
  newdata,
  newtruth = NULL,
  freq = 1000,
  record = FALSE,
  plot = FALSE
)
```

## Arguments

| | |
|---|---|
| `net` | an object of class network, see ?network |
| `param` | vector of trained parameters from the network, see ?train |
| `newdata` | input data to be predicted, a list of vectors (i.e. ragged array) |
| `newtruth` | the truth, a list of vectors to compare with output from the feed-forward network |

| | |
|---|---|
| freq | frequency to print progress updates to the console, default is every 1000th training point |
| record | logical, whether to record details of the prediction. Default is FALSE |
| plot | locical, whether to produce diagnostic plots. Default is FALSE |

## Value

if record is FALSE, the output of the neural network is returned. Otherwise a list of objects is returned including: rec, the predicted probabilities; err, the L1 error between truth and prediction; pred, the predicted categories based on maximum probability; pred_MC, the predicted categories based on maximum probability; truth, the object newtruth, turned into an integer class number

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

NNpredict.regression, network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

## Examples

```
# Example in context:


download_mnist("mnist.RData") # only need to download once
load("mnist.RData") # loads objects train_set, truth, test_set and test_truth

net <- network( dims = c(784,16,16,10),
                activ=list(ReLU(),ReLU(),softmax()))

netwts <- train(dat=train_set,
                truth=truth,
                net=net,
                eps=0.001,
                tol=0.8, # normally would use a higher tol here e.g. 0.95
                loss=multinomial(),
                batchsize=100)

pred <- NNpredict(  net=net,
                    param=netwts$opt,
                    newdata=test_set,
```

```
                    newtruth=test_truth,
                    record=TRUE,
                    plot=TRUE)


# Example 2

N <- 1000
d <- matrix(rnorm(5*N),ncol=5)

fun <- function(x){
    lp <- 2*x[2]
    pr <- exp(lp) / (1 + exp(lp))
    ret <- c(0,0)
    ret[1+rbinom(1,1,pr)] <- 1
    return(ret)
}

d <- lapply(1:N,function(i){return(d[i,])})

truth <- lapply(d,fun)

net <- network( dims = c(5,10,2),
                activ=list(ReLU(),softmax()))

netwts <- train( dat=d,
                 truth=truth,
                 net=net,
                 eps=0.01,
                 tol=100,            # run for 100 iterations
                 batchsize=10,       # note this is not enough
                 loss=multinomial(), # for convergence
                 stopping="maxit")

pred <- NNpredict(  net=net,
                    param=netwts$opt,
                    newdata=d,
                    newtruth=truth,
                    record=TRUE,
                    plot=TRUE)
```

---

NNpredict.regression *NNpredict.regression function*

---

## Description

A function to produce predictions from a trained network

## Usage

```
NNpredict.regression(
  net,
  param,
  newdata,
  newtruth = NULL,
  freq = 1000,
  record = FALSE,
  plot = FALSE
)
```

## Arguments

| | |
|---|---|
| net | an object of class network, see ?network |
| param | vector of trained parameters from the network, see ?train |
| newdata | input data to be predicted, a list of vectors (i.e. ragged array) |
| newtruth | the truth, a list of vectors to compare with output from the feed-forward network |
| freq | frequency to print progress updates to the console, default is every 1000th training point |
| record | logical, whether to record details of the prediction. Default is FALSE |
| plot | locical, whether to produce diagnostic plots. Default is FALSE |

## Value

if record is FALSE, the output of the neural network is returned. Otherwise a list of objects is returned including: rec, the predicted probabilities; err, the L1 error between truth and prediction; pred, the predicted categories based on maximum probability; pred_MC, the predicted categories based on maximum probability; truth, the object newtruth, turned into an integer class number

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

NNpredict, network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

`no_regularisation` *no_regularisation function*

## Description

A function to return the no regularisation strategy for a network object.

## Usage

```
no_regularisation()
```

## Value

list containing functions to evaluate the cost modifier and grandient modifier

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, L1_regularisation, L2_regularisation

## Examples

```
# Example in context: NOTE with the network function
# no_regularisation() is the default, so this argument
# actually need not be included

net <- network( dims = c(784,16,16,10),
                regulariser = no_regularisation(),
                activ=list(ReLU(),logistic(),softmax()))
```

Qloss *Qloss function*

## Description

A function to evaluate the quadratic loss function and the derivative of this function to be used when training a neural network.

## Usage

```
Qloss()
```

## Value

a list object with elements that are functions, evaluating the loss and the derivative

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, multinomial, no_regularisation, L1_regularisation, L2_regularisation

## Examples

```
# Example in context:


netwts <- train(dat=train_set,
                truth=truth,
                net=net,
                eps=0.001,
                tol=0.95,
                loss=Qloss(), # note Qloss is actually the default
                batchsize=100)
```

ReLU *ReLU function*

## Description

A function to evaluate the ReLU activation function, the derivative and cost derivative to be used in defining a neural network.

## Usage

```
ReLU()
```

## Value

a list of functions used to compute the activation function, the derivative and cost derivative.

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, smoothReLU, ident, softmax

## Examples

```
# Example in context

net <- network( dims = c(100,50,20,2),
                activ=list(ReLU(),ReLU(),softmax()))
```

smoothReLU *smoothReLU function*

## Description

A function to evaluate the smooth ReLU (AKA softplus) activation function, the derivative and cost derivative to be used in defining a neural network.

## Usage

```
smoothReLU()
```

## Value

a list of functions used to compute the activation function, the derivative and cost derivative.

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, ident, softmax

## Examples

```
# Example in context

net <- network( dims = c(100,50,20,2),
                activ=list(smoothReLU(),ReLU(),softmax()))
```

`softmax` *softmax function*

## Description

A function to evaluate the softmax activation function, the derivative and cost derivative to be used in defining a neural network. Note that at present, this unit can only be used as an output unit.

## Usage

```
softmax()
```

## Value

a list of functions used to compute the activation function, the derivative and cost derivative.

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident

## Examples

```
# Example in context

net <- network( dims = c(100,50,20,2),
                activ=list(logistic(),ReLU(),softmax()))
```

## stopping *stopping function*

### Description

Generic function for implementing stopping methods

### Usage

```
stopping(...)
```

### Arguments

| | |
|---|---|
| ... | additional arguments |

### Value

method stopping

### See Also

stopping.default, stopping.maxit

## stopping.default *stopping.default function*

### Description

A function to halt computation when curcost < tol

### Usage

```
## Default S3 method:
stopping(cost, curcost, count, tol, ...)
```

### Arguments

| | |
|---|---|
| cost | the value of the loss function passed in |
| curcost | current measure of cost, can be different to the parameter 'cost' above e.g. may consider smoothed cost over the last k iterations |
| count | iteration count |
| tol | tolerance, or limit |
| ... | additional arguments |

### Value

...

### See Also

stopping.maxit

---

`stopping.maxit` *stopping.maxit function*

---

### Description

A function to halt computation when the number of iterations reaches a given threshold, tol

### Usage

```
## S3 method for class 'maxit'
stopping(cost, curcost, count, tol, ...)
```

### Arguments

| | |
|---|---|
| `cost` | the value of the loss function passed in |
| `curcost` | current measure of cost, can be different to the parameter 'cost' above e.g. may consider smoothed cost over the last k iterations |
| `count` | iteration count |
| `tol` | tolerance, or limit |
| `...` | additional arguments |

### Value

...

---

`stopping.revdir` *stopping.revdir function*

---

### Description

A function to halt computation when curcost > tol

### Usage

```
## S3 method for class 'revdir'
stopping(cost, curcost, count, tol, ...)
```

### Arguments

| | |
|---|---|
| cost | the value of the loss function passed in |
| curcost | current measure of cost, can be different to the parameter 'cost' above e.g. may consider smoothed cost over the last k iterations |
| count | iteration count |
| tol | tolerance, or limit |
| ... | additional arguments |

### Value

...

### See Also

stopping.maxit

---

## train *train function*

---

### Description

A function to train a neural network defined using the network function.

### Usage

```
train(
  dat,
  truth,
  net,
  loss = Qloss(),
  tol = 0.95,
  eps = 0.001,
  batchsize = NULL,
  dropout = dropoutProbs(),
  parinit = function(n) {     return(runif(n, -0.01, 0.01)) },
  monitor = TRUE,
  stopping = "default",
  update = "classification"
)
```

## Arguments

| | |
|---|---|
| dat | the input data, a list of vectors |
| truth | the truth, a list of vectors to compare with output from the feed-forward network |
| net | an object of class network, see ?network |
| loss | the loss function, see ?Qloss and ?multinomial |
| tol | stopping criteria for training. Current method monitors the quality of randomly chosen predictions from the data, terminates when the mean predictive probabilities of the last 20 randomly chosen points exceeds tol, default is 0.95 |
| eps | stepsize scaling constant in gradient descent, or stochastic gradient descent |
| batchsize | size of minibatches to be used with stochastic gradient descent |
| dropout | optional list of dropout probabilities ?dropoutProbs |
| parinit | a function of a single parameter returning the initial distribution of the weights, default is uniform on (-0.01,0.01) |
| monitor | logical, whether to produce learning/convergence diagnostic plots |
| stopping | method for stopping computation default, 'default', calls the function stopping.default |
| update | and default for meth is 'classification', which calls updateStopping.classification |

## Value

optimal cost and parameters from the trained network; at present, diagnostic plots are produced illustrating the parameters of the model, the gradient and stopping criteria trace.

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

## Examples

```
# Example in context:


download_mnist("mnist.RData") # only need to download once
load("mnist.RData") # loads objects train_set, truth, test_set and test_truth
```

```
net <- network( dims = c(784,16,16,10),
                activ=list(ReLU(),ReLU(),softmax()))

netwts <- train(dat=train_set,
                truth=truth,
                net=net,
                eps=0.001,
                tol=0.8, # normally would use a higher tol here e.g. 0.95
                loss=multinomial(),
                batchsize=100)

pred <- NNpredict(  net=net,
                    param=netwts$opt,
                    newdata=test_set,
                    newtruth=test_truth,
                    record=TRUE,
                    plot=TRUE)


# Example 2

N <- 1000
d <- matrix(rnorm(5*N),ncol=5)

fun <- function(x){
    lp <- 2*x[2]
    pr <- exp(lp) / (1 + exp(lp))
    ret <- c(0,0)
    ret[1+rbinom(1,1,pr)] <- 1
    return(ret)
}

d <- lapply(1:N,function(i){return(d[i,])})

truth <- lapply(d,fun)

net <- network( dims = c(5,10,2),
                activ=list(ReLU(),softmax()))

netwts <- train( dat=d,
                 truth=truth,
                 net=net,
                 eps=0.01,
                 tol=100,            # run for 100 iterations
                 batchsize=10,       # note this is not enough
                 loss=multinomial(), # for convergence
                 stopping="maxit")

pred <- NNpredict(  net=net,
                    param=netwts$opt,
                    newdata=d,
                    newtruth=truth,
                    record=TRUE,
```

```
                    plot=TRUE)
```

---

`updateStopping` *updateStopping function*

---

## Description

Generic function for updating stopping criteria

## Usage

```
updateStopping(...)
```

## Arguments

| | |
|---|---|
| `...` | additional arguments |

## Value

method updateStopping

## See Also

updateStopping.classification, updateStopping.regression

---

`updateStopping.classification` *updateStopping.classification function*

---

## Description

A function to update the stopping criteria for a classification problem.

## Usage

```
## S3 method for class 'classification'
updateStopping(
  dat,
  parms,
  net,
  truth,
  testoutput,
  count,
  monitor,
  mx,
  curcost,
  ...
)
```

### Arguments

| | |
|---|---|
| dat | data object |
| parms | model parameters |
| net | an object of class network |
| truth | the truth, to be compared with network outputs |
| testoutput | a vector, the history of the stopping criteria |
| count | iteration number |
| monitor | logical, whether to produce a diagnostic plot |
| mx | a number to be monitored e.g. the cost of the best performing paramerer configuration to date |
| curcost | current measure of cost, can be different to the value of the loss function e.g. may consider smoothed cost (i.e. loss) over the last k iterations |
| ... | additional arguments |

### Value

curcost, testoutput and mx, used for iterating the maximisation process

---

## updateStopping.regression

*updateStopping.regression function*

---

### Description

A function to update the stopping criteria for a classification problem.

### Usage

```
## S3 method for class 'regression'
updateStopping(
  dat,
  parms,
  net,
  truth,
  testoutput,
  count,
  monitor,
  mx,
  curcost,
  ...
)
```

### Arguments

| | |
|---|---|
| dat | data object |
| parms | model parameters |
| net | an object of class network |
| truth | the truth, to be compared with network outputs |
| testoutput | a vector, the history of the stopping criteria |
| count | iteration number |
| monitor | logical, whether to produce a diagnostic plot |
| mx | a number to be monitored e.g. the cost of the best performing paramerer configuration to date |
| curcost | current measure of cost, can be different to the value of the loss function e.g. may consider smoothed cost (i.e. loss) over the last k iterations |
| ... | additional arguments |

### Value

curcost, testoutput and mx, used for iterating the maximisation process

---

## weights2list *weights2list function*

---

### Description

A function to convert a vector of weights into a ragged array (coded here a list of vectors)

### Usage

```
weights2list(weights, dims)
```

### Arguments

| | |
|---|---|
| weights | a vector of weights |
| dims | the dimensions of the network as stored from a call to the function network, see ?network |

### Value

a list object with appropriate structures for compatibility with the functions network, train, MLP_net and backpropagation_MLP

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/


## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, logistic, ReLU, smoothReLU, ident, softmax, Qloss, multinomial, NNgrad_test, weights2list, bias2list, biasInit, memInit, gradInit, addGrad, nnetpar, nbiaspar, addList, no_regularisation, L1_regularisation, L2_regularisation

---

`wmultinomial` *wmultinomial function*

---

## Description

A function to evaluate the weighted multinomial loss function and the derivative of this function to be used when training a neural network. This is eqivalent to a multinomial cost function employing a Dirichlet prior on the probabilities. Its effect is to regularise the estimation so that in the case where we apriori expect more of one particular category compared to another then this can be included in the objective.

## Usage

```
wmultinomial(w, batchsize)
```

## Arguments

| | |
|---|---|
| `w` | a vector of weights, adding up whose length is equal to the output length of the net |
| `batchsize` | of batch used in inference WARNING: ensure this matches with actual batchsize used! |

## Value

a list object with elements that are functions, evaluating the loss and the derivative

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/

## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, Qloss, no_regularisation, L1_regularisation, L2_regularisation

## Examples

```
netwts <- train(dat=train_set,
                truth=truth,
                net=net,
                eps=0.001,
                tol=0.95,
                loss=wmultinomial(c(10,5,6,9)), # here assuming output of length 4
                batchsize=100)
```

---

`wQloss` *wQloss function*

---

## Description

A function to evaluate the weighted quadratic loss function and the derivative of this function to be used when training a neural network.

## Usage

```
wQloss(w)
```

## Arguments

| | |
|---|---|
| `w` | a vector of weights, adding up to 1, whose length is equalt to the output length of the net |

## Value

a list object with elements that are functions, evaluating the loss and the derivative

## References


1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Francis Bach. Deep Learning. (2016)
2. Terrence J. Sejnowski. The Deep Learning Revolution (The MIT Press). (2018)
3. Neural Networks YouTube playlist by 3brown1blue: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
4. http://neuralnetworksanddeeplearning.com/

## See Also

network, train, backprop_evaluate, MLP_net, backpropagation_MLP, multinomial, no_regularisation, L1_regularisation, L2_regularisation

## Examples

```
# Example in context:


netwts <- train(dat=train_set,
                truth=truth,
                net=net,
                eps=0.001,
                tol=0.95,
                loss=wQloss(c(10,5,6,9)), # here assuming output of length 4
                batchsize=100)
```

# Index