Package 'tagi'

July 23, 2021

Title Tractable Approximate Gaussian Inference in Neural Networks

Version 0.0.0.9000

Author Magali-Chen Goulet [aut, cre],
Mélina Mailhot [aut],
James-Alexandre Goulet [aut],
Luong-Ha Nguyen [aut]

Maintainer Magali-Chen Goulet <mag_goul@live.concordia.ca>

Description In this package, we implement the Tractable Approximate Gaussian Inference (TAGI) is a method developped by Goulet et al. (2020), used in Bayesian neural networks.

License GPL (>= 3)

URL https://github.com/mgoulet847/tagi

BugReports https://github.com/mgoulet847/tagi

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Suggests knitr,
rmarkdown,
mvtnorm,
randtoolbox,
xgboost

VignetteBuilder knitr

Imports matlab,
stats

Depends R (>= 2.10)

R topics documented:

`activationFunIndex` *Assign ID to activation functions*

Description

This function assigns an ID number depending on the type of activation function.

Usage

```
activationFunIndex(funName)
```

Arguments

`funName`	Type of activation function: "tanh", "sigm", "cdf", "relu" or "softplus"

Value

An ID number which corresponds to:

- 1 if `funName` is "tanh"
- 2 if `funName` is "sigm"
- 3 if `funName` is "cdf"
- 4 if `funName` is "relu"
- 5 if `funName` is "softplus"

`backwardHiddenStateUpdate`
Backward hidden states update

Description

This function updates hidden units from responses to input data. It updates $\mu_{Z|y}$ and $\Sigma_{Z|y}$ from the $Z|y$ distribution for a given layer.

Usage

```
backwardHiddenStateUpdate(mz, Sz, mzF, SzF, SzB, Czz, mzB, idx)
```

Arguments

`mz`	Mean vector of units for the current layer μ_Z
`Sz`	Covariance matrix of units for the current layer Σ_Z
`mzF`	Mean vector of units for the next layer μ_{Z+}
`SzF`	Covariance matrix of units for the next layer Σ_{Z+}
`SzB`	Covariance matrix of units for the next layer given y $\Sigma_{Z+\|y}$
`Czz`	Covariance matrix between units of previous and current layers Σ_{ZZ+}
`mzB`	Mean vector of units for the next layer given y $\mu_{Z+\|y}$
`idx`	Indices for the hidden state update step of the current layer

Details

$f(\boldsymbol{z}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu_{Z|y}}, \boldsymbol{\Sigma_{Z|y}})$ where

$\boldsymbol{\mu_{Z|y}} = \boldsymbol{\mu_Z} + \boldsymbol{J_Z}(\boldsymbol{\mu_{Z^+|y}} - \boldsymbol{\mu_{Z^+}})$

$\boldsymbol{\Sigma_{Z|y}} = \boldsymbol{\Sigma_Z} + \boldsymbol{J_Z}(\boldsymbol{\Sigma_{Z^+|y}} - \boldsymbol{\Sigma_{Z^+}})\boldsymbol{J_Z^T}$

$\boldsymbol{J_Z} = \boldsymbol{\Sigma_{ZZ^+}}\boldsymbol{\Sigma_{Z^+}^{-1}}$

Value

- Mean vector of units for the current layer given y $\mu_{Z|y}$
- Covariance matrix of units for the current layer given y $\Sigma_{Z|y}$

`backwardParameterUpdate` *Backward parameters update*

Description

This function updates parameters from responses to input data. It updates $\mu_{\theta|y}$ and $\Sigma_{\theta|y}$ from the $\theta|y$ distribution for a given layer.

Usage

```
backwardParameterUpdate(mp, Sp, mzF, SzF, SzB, Czp, mzB, idx)
```

Arguments

`mp`	Mean vector of parameters for the current layer μ_θ
`Sp`	Covariance matrix of parameters for the current layer Σ_θ
`mzF`	Mean vector of units for the next layer μ_{Z^+}
`SzF`	Covariance matrix of units for the next layer Σ_{Z^+}
`SzB`	Covariance matrix of units for the next layer given y $\Sigma_{Z^+\|y}$
`Czp`	Covariance matrix between units and parameters for the current layer $\Sigma_{\theta Z^+}$
`mzB`	Mean vector of units for the next layer given y $\mu_{Z^+\|y}$
`idx`	Indices for the parameter update step of the current layer

Details

$f(\boldsymbol{\theta}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu_{\theta|y}}, \boldsymbol{\Sigma_{\theta|y}})$ where

$\boldsymbol{\mu_{\theta|y}} = \boldsymbol{\mu_\theta} + \boldsymbol{J_\theta}(\boldsymbol{\mu_{Z^+|y}} - \boldsymbol{\mu_{Z^+}})$

$\boldsymbol{\Sigma_{\theta|y}} = \boldsymbol{\Sigma_\theta} + \boldsymbol{J_\theta}(\boldsymbol{\Sigma_{Z^+|y}} - \boldsymbol{\Sigma_{Z^+}})\boldsymbol{J_\theta^T}$

$\boldsymbol{J_\theta} = \boldsymbol{\Sigma_{\theta Z^+}}\boldsymbol{\Sigma_{Z^+}^{-1}}$

Value

- Mean vector of parameters for the current layer given y $\mu_{\theta|y}$
- Covariance matrix of parameters for the current layer given y $\Sigma_{\theta|y}$

`batchDerivative` *One iteration of the TAGI with derivative calculations*

Description

This function goes through one learning iteration of the neural network model using TAGI with derivative calculations.

Usage

```
batchDerivative(NN, theta, normStat, states, x, Sx, y, dlayer)
```

Arguments

`NN`	Lists the structure of the neural network
`theta`	List of parameters
`normStat`	Normalized statistics
`states`	List of states
`x`	Input data
`Sx`	Variance of input data
`y`	Response data
`dlayer`	Layer from which derivatives will be in respect to

Value

- List of parameters
- List of normalized statistics
- Mean of predicted responses
- Variance of predicted responses
- Mean of first derivative of predicted responses
- Variance of first derivative of predicted responses
- Covariance between derivatives and inputs
- Mean of second derivative of predicted responses

BH *Price of 506 Boston houses.*

Description

This dataset was originally from the StatLib archive. It contains the price and other attributes of 506 Boston houses.

Usage

```
BH
```

Format

A data frame with 506 rows and 14 variables:

CRIM per capita crime rate by town

ZN proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS proportion of non-retail business acres per town

CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

NOX nitric oxides concentration (parts per 10 million)

RM average number of rooms per dwelling

AGE proportion of owner-occupied units built prior to 1940

DIS weighted distances to five Boston employment centres

RAD index of accessibility to radial highways

TAX full-value property-tax rate per $10,000

PTRATIO pupil-teacher ratio by town

B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town

LSTAT % lower status of the population

MEDV median value of owner-occupied homes in $1000's

Details

The dataset from the TAGI repository was used for comparison purposes, but the original dataset was published by Harrison, D. and Rubinfeld, D.L.

Source

https://github.com/CivML-PolyMtl/TAGI/blob/master/BostonHousing/data/BostonHousing.mat

References

http://lib.stat.cmu.edu/datasets/boston

Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978.

buildCzp — *Reformat covariance matrix between units and parameters*

Description

This function properly reformats covariance matrix between units and parameters $\Sigma_{Z\theta}$ for the update step.

Usage

```
buildCzp(Czw, Czb, currenthiddenUnit, prevhiddenUnit, batchSize)
```

Arguments

Czw	Covariance matrix between units and weights for the current layer
Czb	Covariance matrix between units and baises for the current layer
currenthiddenUnit	Number of units in the current layer
prevhiddenUnit	Number of units in the previous layer
batchSize	Number of observations trained at the same time

Value

Reformatted covariance matrix between units and parameters

buildCzz — *Reformat covariance matrix between units of the previous and current layers*

Description

This function properly reformats covariance matrix between units of the previous and current layers Σ_{ZZ^+} for the update step.

Usage

```
buildCzz(Czz, currenthiddenUnit, prevhiddenUnit, batchSize)
```

Arguments

Czz	Covariance matrix between units of the previous and current layers
currenthiddenUnit	Number of units in the current layer
prevhiddenUnit	Number of units in the previous layer
batchSize	Number of observations trained at the same time

Value

Reformatted covariance matrix between of the previous and current layers

catParameters *Concatenate parameters*

Description

Combines in a single column vector each parameter for all layers.

Usage

```
catParameters(mw, Sw, mb, Sb, mwx, Swx, mbx, Sbx)
```

Arguments

mw	Mean of weights for the current layer
Sw	Covariance of weights for the current layer
mb	Mean of biases for the current layer
Sb	Covariance of biases for the current layer
...	Other parameters

Value

- Mean vector of weights for the current layer
- Covariance vector of weights for the current layer
- Mean vector of biases for the current layer
- Covariance vector of biases for the current layer

compressParameters *Compress parameters*

Description

Put together parameters into a list of parameters.

Usage

```
compressParameters(mw, Sw, mb, Sb, mwx, Swx, mbx, Sbx)
```

Arguments

mw	Mean vector of weights for the current layer
Sw	Covariance vector of weights for the current layer
mb	Mean vector of biases for the current layer
Sb	Covariance vector of biases for the current layer
...	Other parameters

Value

List of parameters

compressStates *Compress states*

Description

Put together states into a list of states.

Usage

```
compressStates(mz, Sz, ma, Sa, J, mdxs, Sdxs, mxs, Sxs)
```

Arguments

mz	Mean of units for each layer
Sz	Covariance of units for each layer
ma	Mean of activated units for each layer
Sa	Covariance of activated units for each layer
J	Jacobian
...	Other parameters

Value

List of states

computeError *Compute error*

Description

This function calculates the Root Mean Square Error (RMSE). It takes as input two vectors (or matrices) with one containing the real y's and the other the predicted y's from the model.

Usage

```
computeError(y, ypred)
```

Arguments

y	Response data
ypred	Mean of predicted responses

Value

RMSE for the given data

covariance *Indices for covariances in the neural network*

Description

This function assigns indices for all covariance elements in the neural network.

Usage

```
covariance(NN)
```

Arguments

NN	Lists the structure of the neural network

Value

NN with new elements:

- Indices (weights and activation units) for deterministic matrix F * μ_{WA} for each layer
- Bias indices for deterministic matrix F * μ_B for each layer
- Indices (weights and activation units) for deterministic matrix F * Σ_{ZWA} for each layer
- Indices for the parameter update step for each layer
- Indices for the hidden state update step for each layer
- Indices (weights and activation units) for deterministic matrix F * $\Sigma_{WA\theta}$ for each layer
- Indices for activation unit for each layer
- Bias indices for deterministic matrix F * Σ_B for each layer

covarianceCzp *Covariance matrices between units and parameters*

Description

This function calculate the covariance matrices between units and parameters Σ_{ZW} and Σ_{ZB} for a given layer.

Usage

```
covarianceCzp(ma, Sp, idxFCwwa, idxFCb)
```

Arguments

ma	Mean vector of activation units from previous layer μ_A
Sp	Covariance matrix of parameters for the current layer Σ_θ
idxFCwwa	Indices for weights and for activation units for the current and previous layers respectively
idxFCb	Indices for biases of the current layer

Value

- Covariance matrix between units and biases for the current layer Σ_{ZB}
- Covariance matrix between units and weights for the current layer Σ_{ZW}

---

covarianceCzz *Covariance matrix between units of the previous and current layers*

Description

This function calculate the covariance matrix between units of the previous and current layers Σ_{ZZ^+} for a given layer.

Usage

```
covarianceCzz(mp, Sz, J, idxCawa)
```

Arguments

mp	Mean vector of parameters for the current layer μ_θ
Sz	Covariance matrix of units for the current layer Σ_Z
J	Jacobian matrix evaluated at μ_Z
idxCawa	Indices for weights and for activation units for the current and previous layers respectively

Value

Covariance matrix between units of previous and current layers Σ_{ZZ^+}

covarianceSa *Calculate variance of activated units*

Description

This function uses lineratization to estimate the covariance matrix of activation units Σ_A.

Usage

```
covarianceSa(J, Sz)
```

Arguments

J	Jacobian matrix evaluated at μ_Z
Sz	Covariance matrix of units for the current layer Σ_Z

Value

The activation units covariance matrix Σ_A

covarianceSz *Covariance matrix of units*

Description

This function calculate the covariance matrix of the units Σ_Z for a given layer.

Usage

```
covarianceSz(mp, ma, Sp, Sa, idxFSwaF, idxFSwaFb)
```

Arguments

mp	Mean vector of parameters for the current layer
ma	Mean vector of activation units from previous layer
Sp	Covariance matrix of parameters for the current layer
Sa	Covariance matrix of activation units from previous layer
idxFSwaF	Indices for weights and for activation units for the current and previous layers respectively
idxFSwaFb	Indices for biases of the current layer

Value

Covariance matrix of units for the current layer Σ_Z

covdx *Covariance between derivatives and hidden states*

Description

This function calculates covariance between derivatives and hidden states. It is not related to the derivative calculation process. It could be used infer Z (hidden states) with the constraint that the derivative of g with respect to Z equals 0.

Usage

```
covdx(mwo, mw, mdgo2, mpdi, mdgoe, Cdozi, Cdizi, ni, no, no2, B)
```

Arguments

mwo	Mean vector of weights for the next layer
mw	Mean vector of weights for the current layer
mdgo2	Mean vector of product of derivatives in second next layer
mpdi	Mean vector of first derivative product wd of current layer
mdgoe	Mean of product of derivatives at each node in next layer
Cdozi	Covariance between derivative (next) and hidden (current) layers
Cdizi	Covariance between derivative and hidden layers (same layer)
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size

Value

Covariance between derivative and hidden states

createDevCellarray *States initialization (unit matrices)*

Description

Initiliazes neural network derivative states at 1.

Usage

```
createDevCellarray(nodes, numlayers, B, rB)
```

Arguments

nodes	Vector which contains the number of nodes at each layer
numlayers	Number of layers in the neural network
B	Batch size
rB	Number of times batch size is repeated

Value

Unit matrices for each layer

`createInitCellwithArray` *Initialization (matrix of lists)*

Description

Initializes a matrix containing lists.

Usage

```
createInitCellwithArray(numlayers)
```

Arguments

`numlayers`	Number of layers in the neural network

Value

Matrix containing empty lists

`createStateCellarray` *States initialization (zero-matrices)*

Description

Initiliazes neural network states at 0.

Usage

```
createStateCellarray(nodes, numlayers, B, rB)
```

Arguments

`nodes`	Vector which contains the number of nodes at each layer
`numlayers`	Number of layers in the neural network
`B`	Batch size
`rB`	Number of times batch size is repeated

Value

Zero-matrices for each layer

denormalize *Denormalize data*

Description

This function denormalizes response data processed by the neural network.

Usage

```
denormalize(yn, syn, myntrain, syntrain)
```

Arguments

yn	Predicted responses
syn	Variance of the predicted responses
myntrain	Mean vector of responses from training set
syntrain	Variance vector of responses from training set

Value

- Mean of denormalized predicted responses
- Variance of denormalized predicted responses

derivative *Derivative calculation*

Description

This function does derivative calculations.

Usage

```
derivative(NN, theta, states, mda, Sda, mdda, Sdda, dlayer)
```

Arguments

NN	Lists the structure of the neural network
theta	List of parameters
states	List of states
mda	Mean vectors of activation units' first derivative
Sda	Covariance matrices of activation units' first derivative
mdda	Mean vectors of activation units' second derivative
Sdda	Covariance matrices of activation units' second derivative
dlayer	layer from which derivatives will be in respect to

Value

- Mean of first derivative of predicted responses
- Variance of first derivative of predicted responses
- Covariance between derivatives and inputs
- Mean of second derivative of predicted responses

`extractParameters` *Extract parameters*

Description

Extract parameters from list of parameters.

Usage

```
extractParameters(theta)
```

Arguments

`theta`	List of parameters

Value

- Mean vector of weights for the current layer
- Covariance vector of weights for the current layer
- Mean vector of biases for the current layer
- Covariance vector of biases for the current layer

`extractStates` *Extract states*

Description

Extract states from list of states.

Usage

```
extractStates(states)
```

Arguments

`states`	List of states

Value

- Mean of units for each layer
- Covariance of units for each layer
- Mean of activated units for each layer
- Covariance of activated units for each layer
- Jacobian

`fcCombinaisonDnode` *Combination of products of first derivative (iterations on nodes)*

Description

This function calculates mean of combination of products of first derivatives (wd)*(wd). Each node is multiplied to another node in the same layer (including itself). Their weights are both pointing to the same node in the next layer.

Usage

```
fcCombinaisonDnode(mpdi, mw, Sw, mda, Sda, ni, no, B)
```

Arguments

`mpdi`	Mean vector of first derivative product wd of current layer
`mw`	Mean vector of weights for the current layer
`Sw`	Covariance of weights for the current layer
`mda`	Mean vector of activation units' first derivative from current layer
`Sda`	Covariance of activation units' first derivative from current layer
`ni`	Number of units in current layer
`no`	Number of units in next layer
`B`	Batch size

Value

Mean array of combination of products of first derivatives

fcCombinaisonDweight *Combination of products of first derivative (iterations on weights)*

Description

This function calculates mean of combination of products of first derivatives (wd)*(wd). Each weight is multiplied to another weight (including itself) from the same node. Each node is multiplied to the same node (in the same layer).

Usage

```
fcCombinaisonDweight(mpdi, mw, Sw, mda, Sda, ni, no, B)
```

Arguments

mpdi	Mean vector of first derivative product wd of current layer
mw	Mean vector of weights for the current layer
Sw	Covariance of weights for the current layer
mda	Mean vector of activation units' first derivative from current layer
Sda	Covariance of the activation units' first derivative from current layer
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

Mean array of combination of products of first derivatives

fcCombinaisonDweightNode *Combination of squared products of first derivative*

Description

This function calculates mean of squared products of first derivatives (wd)^2. Every products (weight times node) from current layer are considered which results in a (B*ni x no)-matrix.

Usage

```
fcCombinaisonDweightNode(mpdi, mw, Sw, mda, Sda, ni, no, B)
```

Arguments

mpdi	Mean vector of first derivative product wd of current layer
mw	Mean vector of weights for the current layer
Sw	Covariance of weights for the current layer
mda	Mean vector of activation units' first derivative from current layer
Sda	Covariance of activation units' first derivative from current layer
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

Mean matrix of squared products of first derivatives

---

fcCombinaisonDweightNodeAll

All possible combinations of products of first derivatives

Description

This function calculates mean of products of first derivatives wd*wd. Since both weight and node are iterated over all products, every products (weight times node) from current layer are considered which results in a (Bni x no x noni)-array. I.e. each dimension of the array represents a single product being multiplied to all other possible products from current layer. Order is as followed: w11d1, w12d2, w13d3, ..., w1nidni, w21d1, w22d2, ..., w2nidni, ... wno1d1, ..., wnonidni

Usage

```
fcCombinaisonDweightNodeAll(mpdi, mpdin, mpdiw, ni, no, B)
```

Arguments

mpdi	Mean vector of first derivative product wd of current layer
mpdin	Mean array of combination of products of first derivatives (iterations on nodes)
mpdiw	Mean array of combination of products of first derivatives (iterations on weights)
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

Mean array of combination of products of first derivatives

`fcCovaddddddw` *Covariance between first and second derivatives from consecutive layers*

Description

This function calculates covariance between weights and second derivatives and covariance between first and second derivatives from consecutive layers.

Usage

```
fcCovaddddddw(mao, mai, mdao, Caoai, Cdodi, Caow, Cdow, acto, acti, ni, no, B)
```

Arguments

`mao`	Mean vector of activation units from next layer
`mai`	Mean vector of activation units from current layer
`mdao`	Mean vector of activation units' first derivative from next layer
`Caoai`	Covariance between activation units from current and next layers
`Cdodi`	Covariance between first derivatives from current and next layers
`Caow`	Covariance between activation units and weights
`Cdow`	Covariance between derivatives and weights
`acto`	Activation function index for next layer defined by `activationFunIndex`
`acti`	Activation function index for current layer defined by `activationFunIndex`
`ni`	Number of units in current layer
`no`	Number of units in next layer
`B`	Batch size

Value

- Covariance between first and second derivatives from consecutive layers
- Covariance between second derivatives from next layer and weights

`fcCovawaa` *Covariance between activation units and weights*

Description

This function calculates covariance between activation units and weights and covariance between activation units from consecutive layers.

Usage

```
fcCovawaa(mw, Sw, Jo, mai, Sai, ni, no, B)
```

Arguments

mw	Mean vector of weights for the current layer
Sw	Covariance of weights for the current layer
Jo	Jacobian of next layer
mai	Mean vector of activation units from current layer
Sai	Covariance of activation units from current layer
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

- Covariance between activation units and weights
- Covariance between activation units from current and next layers

---

fcCovaz *Covariance between activation and hidden units*

Description

This function calculates covariance between activation and hidden units.

Usage

```
fcCovaz(Jo, J, Sz, mw, ni, no, B)
```

Arguments

Jo	Jacobian of next layer
J	Jacobian of current layer
Sz	Covariance of units from current layer
mw	Mean vector of weights for the current layer
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

- Covariance between activation and hidden layers (same layer)
- Covariance between activation (next) and hidden (current) layers

fcCovdaddd — *Covariance between first and second derivatives from consecutive layers*

Description

This function calculates covariance between activation units and first derivatives and covariance between first and second derivatives from consecutive layers.

Usage

```
fcCovdaddd(mao, mai, mdai, Caoai, Cdodi, acti, ni, no, B)
```

Arguments

mao	Mean vector of activation units from next layer
mai	Mean vector of activation units from current layer
mdai	Mean vector of activation units' first derivative from current layer
Caoai	Covariance between activation units from current and next layers
Cdodi	Covariance between derivatives from current and next layers
acti	Activation function index for current layer defined by activationFunIndex
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

- Covariance between first derivatives from next layer and activation units from current layer
- Covariance between first and second derivatives from consecutive layers

fcCovDlayer — *Covariance between products of derivatives and weights*

Description

This function calculates covariance between products of derivatives and weights from consecutive layers.

Usage

```
fcCovDlayer(mdgo2, mwo, Cdowdi, ni, no, no2, B)
```

Arguments

mdgo2	Mean vector of product of derivatives in second next layer
mwo	Mean vector of weights for the next layer
Cdowdi	Covariance between derivatives and weights times derivatives
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size

Value

Covariance between weights times derivatives from consecutive layers

---

fcCovdwd *Covariance between derivatives and weights*derivatives*

Description

This function calculates covariance between derivatives and weights and covariance between derivatives from consecutive layers.

Usage

```
fcCovdwd(md, mw, Cdow, Cdodi, ni, no, B)
```

Arguments

md	Mean vector of derivatives
mw	Mean vector of weights for the current layer
Cdow	Covariance between derivatives and weights
Cdodi	Covariance between derivatives from current and next layers
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

Covariance between derivatives and weights times derivatives

fcCovdwddd *Covariance between derivatives and weights*

Description

This function calculates covariance between derivatives and weights and covariance between derivatives from consecutive layers.

Usage

```
fcCovdwddd(mao, Sao, mai, Sai, Caow, Caoai, acto, acti, ni, no, B)
```

Arguments

mao	Mean vector of activation units from next layer
Sao	Covariance of activation units from next layer
mai	Mean vector of activation units from current layer
Sai	Covariance of activation units from current layer
Caow	Covariance between activation units and weights
Caoai	Covariance between activation units from current and next layers
acto	Activation function index for next layer defined by activationFunIndex
acti	Activation function index for current layer defined by activationFunIndex
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

- Covariance between derivatives and weights
- Covariance between derivatives from current and next layers

fcCovdz *Covariance between derivatives and hidden Units*

Description

This function calculates covariance between derivatives and hidden units.

Usage

```
fcCovdz(mao, mai, Caizi, Caozi, acto, acti, ni, no, B)
```

Arguments

mao	Mean vector of activation units from next layer
mai	Mean vector of activation units from current layer
Caizi	covariance between activation and hidden layers (same layer)
Caozi	covariance between activation (next layer) and hidden (current) layers
acto	Activation function index for next layer defined by activationFunIndex
acti	Activation function index for current layer defined by activationFunIndex
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

- Covariance between derivative (next) and hidden (current) layers
- Covariance between derivative and hidden layers (same layer)

---

fcCovwdo2wdiwdi *Covariance between products in (same) next and current layers*

Description

This function calculates covariance cov(wdo^2,wdiwdi) of weights times derivatives products terms when there are two products in both next and current layers. The product fom next layer is the same squared.

Usage

```
fcCovwdo2wdiwdi(mpdo, Cwdowdiwdi)
```

Arguments

mpdo	Mean vector of first derivative product wd of next layer
Cwdowdiwdi	Covariance cov(wdo,wdi*wdi) of weights times derivatives products terms when there are one product in next layer and two in current

Value

Covariance cov(wdo^2,wdi*wdi) of weights times derivatives products terms when there are two products in both next and current layers

fcCovwdowdi2 — *Covariance between next layer product and current layer squared product*

Description

This function calculates covariance cov(wdo,(wdi)^2) of weights times derivatives products terms when there are one product in next layer and two squared in current.

Usage

```
fcCovwdowdi2(mpdi, Cdgoddgik)
```

Arguments

mpdi	Mean vector of first derivative product wd of current layer
Cdgoddgik	Covariance between weights times derivatives from consecutive layers

Value

Covariance cov(wdo,(wdi)^2) of weights times derivatives products terms when there is one product in next layer and two squared in current

fcCovwdowdiwdi — *Covariance between next layer product and current layer multiplied products*

Description

This function calculates covariance cov(wdo,wdiwdi) of weights times derivatives products terms when there are one product in next layer and two in current.

Usage

```
fcCovwdowdiwdi(mpdi, Cdgoddgik, ni, no, B)
```

Arguments

mpdi	Mean vector of first derivative product wd of current layer
Cdgoddgik	Covariance between weights times derivatives from consecutive layers
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

Covariance cov(wdo,wdi*wdi) of weights times derivatives products terms when there is one product in next layer and two in current

`fcCwdowdowdiwdi` *Covariance between next layer multiplied products and current layer multiplied products (minimum 3 hidden layers)*

Description

This function calculates covariance cov(wdowdo,wdiwdi) where all terms can be different. It is used when there are at least 3 hidden layers and second next layer is a product of only two terms (wdo2).

Usage

```
fcCwdowdowdiwdi(mpdi, mpdo, Cdgodgi, acti, ni, no, no2, B)
```

Arguments

`mpdi`	Mean vector of first derivative product wd of current layer
`mpdo`	Mean vector of first derivative product wd of next layer
`Cdgodgi`	Covariance between weights times derivatives from consecutive layers
`acti`	Activation function index for current layer defined by `activationFunIndex`
`ni`	Number of units in current layer
`no`	Number of units in next layer
`no2`	Number of units in second next layer
`B`	Batch size

Value

Covariance cov(wdowdo,wdiwdi) where all terms can be different

`fcCwdowdowdiwdi_4hl` *Covariance between next layer multiplied products and current layer multiplied products (minimum 4 hidden layers)*

Description

This function calculates covariance cov(wdowdo,wdiwdi) where all terms can be different. It is used when there are at least 4 hidden layers.

Usage

```
fcCwdowdowdiwdi_4hl(mpdi, mpdo, mdgo2, Cdgodgi, acti, ni, no, no2, B)
```

Arguments

mpdi	Mean vector of first derivative product wd of current layer
mpdo	Mean vector of first derivative product wd of next layer
mdgo2	Mean vector of product of derivatives in second next layer
Cdgodgi	Covariance between weights times derivatives from consecutive layers
acti	Activation function index for current layer defined by activationFunIndex
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size

Value

Covariance cov(wdowdo,wdiwdi) where all terms can be different

---

fcCwdowdowwdi2 *Covariance between next layer multiplied products and current layer multiplied products (same derivative)*

Description

This function calculates covariance cov(wdowdo,wdiwdi) where the di terms are the same, when next second layer involves only a product term (wddo2).

Usage

```
fcCwdowdowwdi2(mpdi, mpdo, Cdgodgi, acti, ni, no, no2, B)
```

Arguments

mpdi	Mean vector of first derivative product wd of current layer
mpdo	Mean vector of first derivative product wd of next layer
Cdgodgi	Covariance between weights times derivatives from consecutive layers
acti	Activation function index for current layer defined by activationFunIndex
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size

Value

Covariance cov(wdowdo,wdiwdi) where the di terms are the same

fcCwdowdowwdi2_3hl *Covariance between next layer multiplied products and current layer multiplied products (same derivative, minimum 3 hidden layers)*

Description

This function calculates covariance cov(wdowdo,wdiwdi) where the di terms are the same when next second layer involves multiplied terms (wdo2wdo2). It is used when there are at least 3 hidden layers.

Usage

```
fcCwdowdowwdi2_3hl(mpdi, mpdo, mdgo2, Cdgodgi, acti, ni, no, no2, B)
```

Arguments

mpdi	Mean vector of first derivative product wd of current layer
mpdo	Mean vector of first derivative product wd of next layer
mdgo2	Mean vector of product of derivatives in second next layer
Cdgodgi	Covariance between weights times derivatives from consecutive layers
acti	Activation function index for current layer defined by activationFunIndex
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size

Value

Covariance cov(wdowdo,wdiwdi) where the di terms are the same

fcDerivative *Derivatives for fully connected layers*

Description

This function calculates mean and variance of derivatives and covariance of derivative and input layers.

Usage

```
fcDerivative(
  mw,
  Sw,
  mwo,
  Jo,
  J,
  mao,
```

```
  Sao,
  mai,
  Sai,
  Szi,
  mdai,
  Sdai,
  mdgo,
  mdgoe,
  Sdgo,
  mdgo2,
  acto,
  acti,
  ni,
  no,
  no2,
  B
)
```

Arguments

mw	Mean vector of weights for the current layer
Sw	Covariance of weights for the current layer
mwo	Mean vector of weights for the next layer
Jo	Jacobian of next layer
J	Jacobian of current layer
mao	Mean vector of activation units from next layer
Sao	Covariance of activation units from next layer
mai	Mean vector of activation units from current layer
Sai	Covariance of activation units from current layer
Szi	Covariance of units from current layer
mdai	Mean vector of activation units' first derivative from current layer
Sdai	Covariance of activation units' first derivative from current layer
mdgo	Mean vector of product of derivatives in next layer
mdgoe	Mean of product of derivatives at each node in next layer
Sdgo	Variance of first derivatives in next layer
mdgo2	Mean vector of product of derivatives in second next layer
acto	Activation function index for next layer defined by activationFunIndex
acti	Activation function index for current layer defined by activationFunIndex
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size

Value

Mean vector of first derivatives

Covariance matrix of first derivatives

Covariance matrix of first derivative and input layer

Covariance between activation units and weights

Covariance between activation units from current and next layers

Covariance between first derivatives and weights

Covariance between first derivatives from current and next layers

Covariance between first derivatives from next layer and weights times derivatives from current layer

---

`fcDerivative2` *Second derivatives for fully connected layers*

Description

This function calculates mean of product of derivatives, when new product term involves second derivatives (wdd).

Usage

```
fcDerivative2(
  mw,
  mwo,
  mao,
  mai,
  mdai,
  mddai,
  mpddi,
  mdgo,
  mdgo2,
  Caoai,
  Cdow,
  Cdodi,
  acti,
  ni,
  no,
  no2,
  B
)
```

Arguments

`mw`	Mean vector of weights for the current layer
`mwo`	Mean vector of weights for the next layer
`mao`	Mean vector of activation units from next layer
`mai`	Mean vector of activation units from current layer

mdai	Mean vector of activation units' first derivative from current layer
mddai	Mean vector of activation units' second derivative from current layer
mpddi	Mean vector of second derivative product wdd of current layer
mdgo	Mean vector of product of derivatives in next layer
mdgo2	Mean vector of product of derivatives in second next layer
Caoai	Covariance between activation units from current and next layers
Cdow	Covariance between first derivatives and weights
Cdodi	Covariance between first derivatives from current and next layers
acti	Activation function index for current layer defined by activationFunIndex
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size

Value

Mean of product terms for second derivative calculations

`fcDerivative3` *Products of first derivatives multiplied to second derivative for fully connected layers*

Description

This function calculates mean of product of derivatives, when new product term involves product of two first derivatives (wdwd) from the same layer multiplied to second derivatives (wdd) from next layer.

Usage

```
fcDerivative3(
  mw,
  Sw,
  mwo,
  mao,
  mai,
  mdao,
  mdai,
  Sdai,
  mpdi,
  mdgo,
  mdgo2,
  Caow,
  Caoai,
  Cdow,
  Cdodi,
  acto,
```

```
  acti,
  ni,
  no,
  no2,
  B,
  dlayer
)
```

Arguments

mw	Mean vector of weights for the current layer
Sw	Covariance of weights for the current layer
mwo	Mean vector of weights for the next layer
mao	Mean vector of activation units from next layer
mai	Mean vector of activation units from current layer
mdao	Mean vector of activation units' first derivative from next layer
mdai	Mean vector of activation units' first derivative from current layer
Sdai	Covariance of activation units' first derivative from current layer
mpdi	Mean vector of first derivative product wd of current layer
mdgo	Mean vector of product of derivatives in next layer
mdgo2	Mean vector of product of derivatives in second next layer
Caow	Covariance between activation units and weights
Caoai	Covariance between activation units from current and next layers
Cdow	Covariance between first derivatives and weights
Cdodi	Covariance between first derivatives from current and next layers
acto	Activation function index for next layer defined by activationFunIndex
acti	Activation function index for current layer defined by activationFunIndex
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size
dlayer	TRUE if derivatives will be in respect to current layer

Value

Mean of product terms for second derivative calculations

`fcDerivative4` *Products of first derivatives multiplied to products of first derivatives for fully connected layers*

Description

This function calculates mean of product of derivatives, when new product term involves product of two first derivatives (wdwd) from the same layer multiplied to product of two first derivatives (wdwd) from next layer, when second next layer is second derivatives (wdd).

Usage

```
fcDerivative4(
  mw,
  Sw,
  mwo,
  mao,
  mai,
  mdao,
  mdai,
  Sdai,
  mpdo,
  mpdi,
  mdgo,
  mdgo2,
  Cdowdi,
  acto,
  acti,
  ni,
  no,
  no2,
  B,
  dlayer
)
```

Arguments

`mw`	Mean vector of weights for the current layer
`Sw`	Covariance of weights for the current layer
`mwo`	Mean vector of weights for the next layer
`mao`	Mean vector of activation units from next layer
`mai`	Mean vector of activation units from current layer
`mdao`	Mean vector of activation units' first derivative from next layer
`mdai`	Mean vector of activation units' first derivative from current layer
`Sdai`	Covariance of activation units' first derivative from current layer
`mpdo`	Mean vector of first derivative product wd of next layer
`mpdi`	Mean vector of first derivative product wd of current layer
`mdgo`	Mean vector of product of derivatives in next layer

mdgo2	Mean vector of product of derivatives in second next layer
Cdowdi	Covariance between first derivatives from next layer and weights times derivatives from current layer
acto	Activation function index for next layer defined by activationFunIndex
acti	Activation function index for current layer defined by activationFunIndex
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size
dlayer	TRUE if derivatives will be in respect to current layer

Value

Mean of product terms for second derivative calculations

`fcDerivative5` *Products of first derivatives multiplied to products of first derivatives (not only last layer) for fully connected layers*

Description

This function calculates mean of product of derivatives, when new product term involves product of two first derivatives (wdwd) from the same layer multiplied to product of two first derivatives (wdwd) from next and second next layers.

Usage

```
fcDerivative5(
  mw,
  Sw,
  mwo,
  mao,
  mai,
  mdao,
  mdai,
  Sdai,
  mpdo,
  mpdi,
  mdgo,
  mdgo2,
  Cdowdi,
  acto,
  acti,
  ni,
  no,
  no2,
  B,
  dlayer
)
```

Arguments

mw	Mean vector of weights for the current layer
Sw	Covariance of weights for the current layer
mwo	Mean vector of weights for the next layer
mao	Mean vector of activation units from next layer
mai	Mean vector of activation units from current layer
mdao	Mean vector of activation units' first derivative from next layer
mdai	Mean vector of activation units' first derivative from current layer
Sdai	Covariance of activation units' first derivative from current layer
mpdo	Mean vector of first derivative product wd of next layer
mpdi	Mean vector of first derivative product wd of current layer
mdgo	Mean vector of product of derivatives in next layer
mdgo2	Mean vector of product of derivatives in second next layer
Cdowdi	Covariance between derivatives from next layer and weights times derivatives from current layer
acto	Activation function index for next layer defined by activationFunIndex
acti	Activation function index for current layer defined by activationFunIndex
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size
dlayer	TRUE if derivatives will be in respect to current layer

Value

Mean of product terms for second derivative calculations

---

fcHiddenStateBackwardPass

Backpropagation (states' deltas) for fully connected layers (many observations)

Description

This function calculates units' deltas at a given layer when using more than one observation at the time.

Usage

```
fcHiddenStateBackwardPass(Sz, Sxs, J, mw, deltaM, deltaS, ni, no, B, rB)
```

Arguments

Sz	Covariance of units from current layer
Sxs	Null by default (not used yet)
J	Jacobian of current layer
mw	Mean vector of weights for the current layer
deltaM	Delta of mean vector of the next layer units given y $\mu_Z \vert y$
deltaS	Delta of covariance matrix of the next layer units given y $\Sigma_Z \vert y$
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size
rB	Number of times batch size is repeated

Value

- Delta of mean vector of current layer units given y $\mu_Z \vert y$
- Delta of covariance matrix of current layer units given y $\Sigma_Z \vert y$

---

fcHiddenStateBackwardPassB1

Backpropagation (states' deltas) for fully connected layers (one observation)

Description

This function calculates units' deltas at a given layer when using one observation at the time.

Usage

```
fcHiddenStateBackwardPassB1(Sz, Sxs, J, mw, deltaM, deltaS, ni, no)
```

Arguments

Sz	Covariance of the units from current layer
Sxs	Null by default (not used yet)
J	Jacobian of current layer
mw	Mean vector of weights for the current layer
deltaM	Delta of mean vector of next layer units given y $\mu_Z \vert y$
deltaS	Delta of covariance matrix of next layer units given y $\Sigma_Z \vert y$
ni	Number of units in current layer
no	Number of units in next layer

Value

- Delta of mean vector of current layer units given y $\mu_Z \vert y$
- Delta of covariance matrix of current layer units given y $\Sigma_Z \vert y$

fcMeanDlayer2array — *Mean of weights times derivatives products terms ((wdo*wdo) x (wwdi^2))*

Description

This function calculates mean of weights times derivatives products terms when adding two of those products from current layer to already calculated expectation that ended with one such product of next layer (i.e. (wdowdo) x (wwdi^2)). Mean terms are in array format. Once added, rows need to be summed to aggregate expectations by node*weight combinations of current layer.

Usage

```
fcMeanDlayer2array(mpdi2w, mdgo, Cwdowdowwdi2, ni, no, B)
```

Arguments

mpdi2w	Combination of products of first derivative of current layer (wd)*(wd) (iterations on weights on the same node)
mdgo	Mean of product of derivatives in next layer
Cwdowdowwdi2	Covariance cov(wdowdo,wdiwdi) of weights times derivatives products terms, where the di terms are the same
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

Mean of weights times derivatives products terms

fcMeanDlayer2row — *Mean of weights times derivatives products terms squared (wdo x (wdi*wdi))*

Description

This function calculates mean of weights times derivatives products terms when adding two of those products from current layer to already calculated expectation that ended with one such product of next layer (i.e. wdo x (wdiwdi)). Mean terms are in array format. Once added, rows need to be summed to aggregate expectations by node*node combinations of current layer.

Usage

```
fcMeanDlayer2row(mpdi, mpdi2, mdgo, Cwdowdiwdi, ni, no, no2, B)
```

Arguments

mpdi	Mean vector of first derivative product wd of current layer
mpdi2	Mean array of combination of products of first derivatives
mdgo	Mean vector of product of derivatives in next layer
Cwdowdiwdi	Covariance cov(wdo,(wdi*wdi)) of weights times derivatives products terms when there is one product in next layer and two in current
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size

Value

Mean of weights times derivatives products terms

---

fcMeanVar *Mean and covariance vectors of units (many observations)*

Description

This function calculate the mean vector of units μ_Z and the covariance matrix of the units $\Sigma_Z$for a given layer.

Usage

```
fcMeanVar(mz, Sz, mw, Sw, mb, Sb, ma, Sa, ni, no, B, rB)
```

Arguments

mw	Mean vector of weights for the current layer
Sw	Covariance of weights for the current layer
mb	Mean vector of biases for the current layer
Sb	Covariance of biases for the current layer
ma	Mean vector of activation units from previous layer
Sa	Covariance of activation units from previous layer
ni	Number of units in previous layer
no	Number of units in current layer
B	Batch size
rB	Number of times batch size is repeated

Value

- Mean vector of units for the current layer μ_Z
- Covariance matrix of units for the current layer Σ_Z

fcMeanVarB1 *Mean and covariance vectors of units (one observation)*

Description

This function calculate the mean vector of units μ_Z and the covariance matrix of the units $\Sigma_Z$for a given layer.

Usage

```
fcMeanVarB1(mw, Sw, mb, Sb, ma, Sa, ni, no)
```

Arguments

mw	Mean vector of weights for the current layer
Sw	Covariance of weights for the current layer
mb	Mean vector of biases for the current layer
Sb	Covariance of biases for the current layer
ma	Mean vector of activation units from previous layer
Sa	Covariance of activation units from previous layer
ni	Number of units in previous layer
no	Number of units in current layer

Value

- Mean vector of units for the current layer μ_Z
- Covariance matrix of units for the current layer Σ_Z

fcMeanVarDlayer *Mean and variance of weights times derivatives products terms*

Description

This function calculates mean and variance of weights times derivatives products terms.

Usage

```
fcMeanVarDlayer(mx, Sx, my, mye, Sy, Cxy, ni, no, no2, B)
```

Arguments

mx	Mean vector of inputs
Sx	Variance of inputs
my	Mean vector of outputs
mye	Mean derivatives at each node in next layer
Sy	Variance of outputs
Cxy	Covariance between inputs and outputs
ni	Number of units in current layer
no	Number of units in next layer
no2	Number of units in second next layer
B	Batch size

Value

- Mean of weights times derivatives products terms
- Covariance between weights times derivatives products terms

---

fcMeanVarDnode *Mean and covariance of derivatives*

Description

This function calculates the mean vector and the covariance matrix for derivatives.

Usage

```
fcMeanVarDnode(mw, Sw, mda, Sda, ni, no, B)
```

Arguments

mw	Mean vector of weights for the current layer
Sw	Covariance of weights for the current layer
mda	Mean vector of activation units' derivative from current layer
Sda	Covariance of activation units' derivative from current layer
ni	Number of units in current layer
no	Number of units in next layer
B	Batch size

Value

- Mean vector of derivatives
- Covariance matrix of derivatives

`fcParameterBackwardPass` *Backpropagation (parameters' deltas) for fully connected layers (many observations)*

Description

This function calculates parameters' deltas at a given layer when using more than one observation at the time.

Usage

```
fcParameterBackwardPass(
  deltaMw,
  deltaSw,
  deltaMb,
  deltaSb,
  Sw,
  Sb,
  ma,
  deltaMr,
  deltaSr,
  ni,
  no,
  B,
  rB
)
```

Arguments

`deltaMw`	next layer delta of mean vector of weights given y $\mu_\theta \vert y$
`deltaSw`	next layer delta of covariance matrix of weights given y $\Sigma_\theta \vert y$
`deltaMb`	next layer delta of mean vector of biases given y $\mu_\theta \vert y$
`deltaSb`	next layer delta of covariance matrix of biases given y $\Sigma_\theta \vert y$
`Sw`	Covariance of weights for the current layer
`Sb`	Covariance of biases for the current layer
`ma`	Mean vector of activation units for the current layer
`deltaMr`	Delta of mean vector of next layer units given y $\mu_Z \vert y$
`deltaSr`	Delta of covariance matrix of next layer units given y $\Sigma_Z \vert y$
`ni`	Number of units in current layer
`no`	Number of units in next layer
`B`	Batch size
`rB`	Number of times batch size is repeated

Value

- Delta of mean vector of weights given y $\mu_\theta|y$
- Delta of covariance matrix of weights given y $\Sigma_\theta|y$
- Delta of mean vector of biases given y $\mu_\theta|y$
- Delta of covariance matrix of biases given y $\Sigma_\theta|y$

`fcParameterBackwardPassB1`

Backpropagation (parameters' deltas) for fully connected layers (one observation)

Description

This function calculates parameters' deltas at a given layer when using one observation at the time.

Usage

```
fcParameterBackwardPassB1(Sw, Sb, ma, deltaMr, deltaSr, ni, no)
```

Arguments

`Sw`	Covariance of weights for the current layer
`Sb`	Covariance of biaises for the current layer
`ma`	Mean vector of activation units for the current layer
`deltaMr`	Delta of mean vector of next layer units given y $\mu_Z|y$
`deltaSr`	Delta of covariance matrix of next layer units given y $\Sigma_Z|y$
`ni`	Number of units in current layer
`no`	Number of units in next layer

Value

- Delta of mean vector of weights given y $\mu_\theta|y$
- Delta of covariance matrix of weights given y $\Sigma_\theta|y$
- Delta of mean vector of biases given y $\mu_\theta|y$
- Delta of covariance matrix of biaises given y $\Sigma_\theta|y$

feedBackward *Backpropagation*

Description

This function feeds the neural network backward from responses to input data.

Usage

```
feedBackward(NN, mp, Sp, mz, Sz, Czw, Czb, Czz, y)
```

Arguments

NN	Lists the structure of the neural network
mp	Mean vectors of parameters for each layer μ_θ
Sp	Covariance matrices of parameters for each layer Σ_θ
mz	Mean vectors of units for each layer μ_Z
Sz	Covariance matrices of units for each layer Σ_Z
Czw	Covariance matrices between units and weights for each layer Σ_{ZW}
Czb	Covariance matrices between units and biases for each layer Σ_{ZB}
Czz	Covariance matrices between previous and current units for each layer Σ_{ZZ^+}
y	Response data

Value

- Updated mean vectors of parameters for each layer μ_θ
- Updated covariance matrices of parameters for each layer Σ_θ

See Also

backwardhiddenStateUpdate, backwardParameterUpdate, forwardhiddenStateUpdate

feedForward *Forward uncertainty propagation*

Description

This function feeds the neural network forward from input data to responses.

Usage

```
feedForward(NN, x, mp, Sp)
```

Arguments

NN	Lists the structure of the neural network
x	Input data
mp	Mean vectors of parameters for each layer μ_θ
Sp	Covariance matrices of parameters for each layer Σ_θ

Value

- Mean vectors of units for each layer μ_Z
- Covariance matrices of units for each layer Σ_Z
- Covariance matrices between units and weights for each layer Σ_{ZW}
- Covariance matrices between units and biases for each layer Σ_{ZB}
- Covariance matrices between previous and current units for each layer Σ_{ZZ^+}

---

feedForwardPass *Forward uncertainty propagation for derivative calculation*

Description

This function feeds the neural network forward from input data to responses and considers components required for derivative calculations.

Usage

```
feedForwardPass(NN, theta, states)
```

Arguments

NN	Lists the structure of the neural network
theta	List of parameters
states	List of states

Value

- Updated states
- Mean vectors of activation units' first derivative
- Covariance matrices of activation units' first derivative
- Mean vectors of activation units' second derivative
- Covariance matrices of activation units' second derivative

forwardHiddenStateUpdate *Last hidden layer states update*

Description

This function updates last hidden layer units using responses. It updates $\mu_{Z^{(0)}|y}$ and $\Sigma_{Z^{(0)}|y}$ from the $Z^{(0)}|y$ distribution.

Usage

```
forwardHiddenStateUpdate(mz, Sz, mzF, SzF, Cyz, y)
```

Arguments

mz	Mean vector of units for the last hidden layer $\mu_{X^{(0)}}$
Sz	Covariance matrix of units for the last hidden layer $\Sigma_{Z^{(0)}}$
mzF	Mean vector of units for the output layer μ_y
SzF	Covariance matrix of tunits for the output layer Σ_y
Cyz	Covariance matrix between last hidden layer units and responses $\Sigma_{YZ^{(0)}}$
y	Response data

Details

$f(\boldsymbol{z}^{(0)}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{z}^{(0)}; \boldsymbol{\mu}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}})$ where

$\boldsymbol{\mu}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}} = \boldsymbol{\mu}_{\boldsymbol{Z}^{(0)}} + \boldsymbol{\Sigma}^{\boldsymbol{T}}_{\boldsymbol{YZ}^{(0)}} \boldsymbol{\Sigma}^{-\mathbf{1}}_{\boldsymbol{Y}} (\boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{Y}})$

$\boldsymbol{\Sigma}_{\boldsymbol{Z}^{(0)}|\boldsymbol{y}} = \boldsymbol{\Sigma}_{\boldsymbol{Z}^{(0)}} - \boldsymbol{\Sigma}^{\boldsymbol{T}}_{\boldsymbol{YZ}^{(0)}} \boldsymbol{\Sigma}^{-\mathbf{1}}_{\boldsymbol{Y}} \boldsymbol{\Sigma}_{\boldsymbol{YZ}^{(0)}}$

Value

- Mean vector of last hidden layer units given y $\mu_{Z^{(0)}|y}$
- Covariance matrix of last hidden layer units given y $\Sigma_{Z^{(0)}|y}$

globalParameterUpdate *Backpropagation (parameters update)*

Description

This function updates parameters.

Usage

```
globalParameterUpdate(theta, deltaTheta)
```

Arguments

theta	List of parameters
deltaTheta	Parameters' deltas (mean and covariance for each)

Value

List of updated parameters

---

hiddenStateBackwardPass

Backpropagation (states' deltas)

Description

This function calculates states' deltas.

Usage

```
hiddenStateBackwardPass(NN, theta, states, y, Sy, udIdx)
```

Arguments

NN	Lists the structure of the neural network
theta	List of parameters
states	List of states
y	Response data
Sy	Variance of responses
udIdx	Specific update IDs

Value

- Delta of mean vector of units given y $\mu_Z|y$ at all layers
- Delta of covariance matrix of units given y $\Sigma_Z|y$ at all layers

initialization *Network initialization*

Description

Verify and add components to the neural network structure.

Usage

```
initialization(NN)
```

Arguments

NN	Lists the structure of the neural network

Value

- NN with all required components
- States of all required elements to perform TAGI

initialization_net *Network initialization*

Description

Verify and add components to the neural network structure.

Usage

```
initialization_net(NN)
```

Arguments

NN	Lists the structure of the neural network

Value

NN with all required components

initializeInputs *Input initialization*

Description

Initializes neural network inputs.

Usage

```
initializeInputs(states, mz0, Sz0, ma0, Sa0, J0, mdxs0, Sdxs0, mxs0, Sxs0, xsc)
```

Arguments

states	States of the neural network
mz0	Input data
Sz0	Variance of input data
ma0	Activated input data
Sa0	Variance of activated input data
J0	Jacobian
...	Other parameters

Value

States of the neural network

initializeStates *States initialization*

Description

Initiliazes neural network states.

Usage

```
initializeStates(nodes, B, rB, xsc)
```

Arguments

nodes	Vector which contains the number of nodes at each layer
B	Batch size
rB	Number of times batch size is repeated

Value

States of the neural network

initializeWeightBias *Weights and biases initialization*

Description

This function initializes the first weights and biases of the neural network.

Usage

```
initializeWeightBias(NN)
```

Arguments

NN	Lists the structure of the neural network

Value

- Initial mean vectors of parameters for each layer
- Initial covariance matrices of parameters for each layer

initializeWeightBiasD *Weights and biases initialization for calculating derivatives*

Description

This function initializes the first weights and biases of the neural network.

Usage

```
initializeWeightBiasD(NN)
```

Arguments

NN	Lists the structure of the neural network

Value

All parameters required in the neural network to perform derivative calculations

innovationVector *Last hidden layer states' deltas update*

Description

This function updates hidden layer units' deltas using next hidden layer' deltas. It updates $\mu_{Z|y}$ and $\Sigma_{Z|y}$ from the $Z|y$ distribution.

Usage

```
innovationVector(SzF, dMz, dSz)
```

Arguments

SzF	Covariance matrix of units for the next layer Σ_y
dMz	Delta of mean vector of units for the next hidden layer μ_Z
dSz	Delta of covariance matrix of units for the next hidden layer Σ_Z

Details

$f(\boldsymbol{z}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu_{Z|y}}, \boldsymbol{\Sigma_{Z|y}})$ where

$\boldsymbol{\mu_{Z|y}} = \boldsymbol{\mu_Z} + \boldsymbol{J_Z}(\boldsymbol{\mu_{Z^+|y}} - \boldsymbol{\mu_{Z^+}})$

$\boldsymbol{\Sigma_{Z|y}} = \boldsymbol{\Sigma_Z} + \boldsymbol{J_Z}(\boldsymbol{\Sigma_{Z^+|y}} - \boldsymbol{\Sigma_{Z^+}})\boldsymbol{J_Z^T}$

$\boldsymbol{J_Z} = \boldsymbol{\Sigma_{ZZ^+}}\boldsymbol{\Sigma_{Z^+}^{-1}}$

Value

- Delta of mean vector of current hidden layer units given y $\mu_Z|y$
- Delta of covariance matrix of current hidden layer units given y $\Sigma_Z|y$

layerEncoder *Layer encoder*

Description

Add layer encoder to the neural network structure.

Usage

```
layerEncoder(NN)
```

Arguments

NN	Lists the structure of the neural network

Value

NN with layer encoder

loglik *Compute log-likelihood*

Description

This function calculates the log-likelihood (LL). It takes as input three vectors (or matrices) with one containing the real y's, one with the predicted y's from the model and the last one with the variance of the y's.

Usage

```
loglik(y, ypred, Vpred)
```

Arguments

y	Response data
ypred	Mean of predicted responses
Vpred	Variance of the predicted responses

Value

LL for the given data

meanA *Calculate mean of activated units*

Description

This function uses lineratization to estimate the activation units mean vector μ_A and the Jacobian matrix evaluated at μ_Z.

Usage

```
meanA(z, mz, funIdx)
```

Arguments

z	Vector of units for the current layer
mz	Mean vector of units for the current layer μ_Z
funIdx	Activation function index defined by activationFunIndex

Value

A list which contains the activation units mean vector μ_A and the Jacobian matrix evaluated at μ_Z

meanMz *Mean vector of units*

Description

This function calculate the mean vector of units μ_Z for a given layer.

Usage

```
meanMz(mp, ma, idxFmwa, idxFmwab)
```

Arguments

mp	Mean vector of parameters for the current layer
ma	Mean vector of activation units from previous layer
idxFmwa	Indices for weights and for activation units for the current and previous layers respectively
idxFmwab	Indices for biases of the current layer

Value

Mean vector of units for the current layer μ_Z

meanVar *Mean, Jacobian and variance of activated units*

Description

This function returns mean vector μ_A, Jacobian matrix evaluated at μ_Z and covariance matrix of activation units Σ_A.

Usage

```
meanVar(z, mz, Sz, funIdx)
```

Arguments

z	Vector of units for the current layer
mz	Mean vector of units for the current layer μ_Z
Sz	Covariance matrix of units for the current layer Σ_Z
funIdx	Activation function index defined by activationFunIndex

Value

- Mean vector of activation units for the current layer μ_A
- Covariance matrix activation units for the current layer Σ_A
- Jacobian matrix evaluated at μ_Z

meanVarDev *Mean and variance of activated units for derivatives*

Description

This function calculates mean vector and covariance matrix of activation units' derivatives.

Usage

```
meanVarDev(mz, Sz, funIdx, bound)
```

Arguments

mz	Mean vector of units for the current layer μ_Z
Sz	Covariance matrix of units for the current layer Σ_Z
funIdx	Activation function index defined by activationFunIndex
bound	If layer is bound

Value

- Mean vector of activation units' first derivative
- Covariance matrix of activation units' first derivative
- Mean vector activation units' second derivative
- Covariance matrix activation units' second derivative

MedicalCost *Medical Cost of 1,338 insureds.*

Description

A dataset containing the medical costs ("charges") and other attributes of 1,338 insureds.

Usage

```
MedicalCost
```

Format

A data frame with 1,338 rows and 10 variables:

age age of the insured

sex gender of the insured, binary (if female)

BMI Body Mass Index of the insured

children number of children covered as dependents

smoker smoking status, binary (if the insured smokes)

region: northeast binary (if the insured lives in that region)

region: southeast binary (if the insured lives in that region)

region: southwest binary (if the insured lives in that region)

region: northwest binary (if the insured lives in that region)

charges medical costs, in US dollars

Details

The original dataset contains 7 variables, but one-hot encoding was used on the "region" categorical variable. It is a dataset that was used in a Kaggle competition.

Source

https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv

---

`network` *One iteration of the Tractable Approximate Gaussian Inference (TAGI)*

Description

This function goes through one learning iteration of the neural network model using TAGI.

Usage

```
network(NN, mp, Sp, x, y)
```

Arguments

`NN`	Lists the structure of the neural network
`mp`	Mean vector of parameters for each layer μ_θ
`Sp`	Covariance matrix of parameters for each layer Σ_θ
`x`	Input data
`y`	Response data

Value

- Updated mean vector of parameters for each layer μ_θ
- Updated covariance matrix of parameters for each layer Σ_θ
- Mean of predicted responses
- Variance of the predicted responses

`normalize` *Normalize data*

Description

This function normalizes data before entering the neural network.

Usage

```
normalize(xtrain, ytrain, xtest, ytest)
```

Arguments

`xtrain`	Training set of input variables
`ytrain`	Training set of responses
`xtest`	Testing set of input variables
`ytest`	Testing set of responses

Value

- Normalized training set of input variables
- Normalized training set of responses
- Normalized testing set of input variables
- Normalized testing set of responses
- Mean vector of input variables from training set
- Covariance matrix of input variables from training set
- Mean vector of responses from training set
- Covariance matrix of responses from training set

`parameterBackwardPass` *Backpropagation (parameters' deltas)*

Description

This function calculates parameter's deltas.

Usage

```
parameterBackwardPass(NN, theta, states, deltaM, deltaS)
```

Arguments

NN	Lists the structure of the neural network
theta	List of parameters
states	List of states
deltaM	Delta of mean vector of units given y $\mu_Z \vert y$ at all layers
deltaS	Delta of covariance matrix of units given y $\Sigma_Z \vert y$ at all layers

Value

Parameters' deltas (mean and covariance for each)

---

parameters *Indices for biases and weights*

Description

This function assigns indices for all weights and biases in the neural network.

Usage

```
parameters(NN)
```

Arguments

NN	List that contains the structure of the neural network

Details

Bias indices are assigned from 1 to the maximum number of biases for a given layer. Then, weight indices start where bias indices end plus one until all weights are assigned an indice. The number of weights for a given layer is the number of units in the previous layer times the number of units in the current one.

For example, if there are 10 units in the previous layer and 50 in the current one, then there would be 50 biases and 500 weights in the current layer. The bias indices would be from 1 to 50 and weight IDs from 51 to 550.

Value

NN with three new elements, each of size (number of layers -1) :

- Weight indices for each layer
- Bias indices for each layer
- Combined weight and bias indices for each layer

regression *Regression problem*

Description

This function trains neural network models to solve a regression problem.

Usage

```
regression(NN, x, y, trainIdx, testIdx)
```

Arguments

NN	Lists the structure of the neural network
x	Input data
y	Response data
trainIdx	Observations IDs that are assigned to the training set
testIdx	Observations IDs that are assigned to the testing set

Value

- Mean vector of parameters for each layer μ_θ
- Covariance matrix of parameters for each layer Σ_θ
- RMSE and LL metrics for each network models created
- Training time of each neural network models created
- Mean of predicted responses
- Variance of the predicted responses

runBatchDerivative *Result of the TAGI with derivative calculations*

Description

This function returns the resulting derivatives from the neural network model using TAGI.

Usage

```
runBatchDerivative(NN, xtrain, ytrain, xtest, ytest)
```

Arguments

NN	Lists the structure of the neural network
xtrain	Training set of input variables
ytrain	Training set of responses
xtest	Testing set of input variables
ytest	Testing set of responses

Value

- Mean of predicted responses
- Variance of the predicted responses
- Mean of first derivative of predicted responses
- Mean of second derivative of predicted responses

`split` *Split data*

Description

This function splits data into training and test sets.

Usage

```
split(x, y, ratio)
```

Arguments

`x`	Input data
`y`	Response data
`ratio`	Training ratio

Value

- Training set of input variables
- Training set of responses
- Testing set of input variables
- Testing set of responses

ToyExample.x_obs *Inputs used in training part for 1D toy problem*

Description

The orignal dataset represents a 1D regression problem from Hernández-Lobato & Adams (2015): $y = x^3 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 9)$ and $x \in [-4, 4]$. In this dataset, x and y are normalized.

Usage

```
ToyExample.x_obs
```

Format

A data frame with 20 rows and 1 variable x

Details

The dataset generated with the seed from the TAGI repository was used for comparison purposes.

Source

https://github.com/CivML-PolyMtl/TAGI/blob/master/ToyExample/ToyExample_1D.m

References

Hernández-Lobato, J. M., & Adams, R. "Probabilistic backpropagation for scalable learning of bayesian neural networks." International Conference on Machine Learning. 2015.

ToyExample.x_val *Inputs used in validation part for 1D toy problem*

Description

The orignal dataset represents a 1D regression problem from Hernández-Lobato & Adams (2015): $y = x^3 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 9)$ and $x \in [-4, 4]$. In this dataset, x and y are normalized.

Usage

```
ToyExample.x_val
```

Format

A data frame with 20 rows and 1 variable x

Details

The dataset generated with the seed from the TAGI repository was used for comparison purposes.

Source

https://github.com/CivML-PolyMtl/TAGI/blob/master/ToyExample/ToyExample_1D.m

References

Hernández-Lobato, J. M., & Adams, R. "Probabilistic backpropagation for scalable learning of bayesian neural networks." International Conference on Machine Learning. 2015.

ToyExample.y_obs *Responses used in training part for 1D toy problem*

Description

The orignal dataset represents a 1D regression problem from Hernández-Lobato & Adams (2015): $y = x^3 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 9)$ and $x \in [-4, 4]$. In this dataset, x and y are normalized.

Usage

```
ToyExample.y_obs
```

Format

A data frame with 20 rows and 1 variable y

Details

The dataset generated with the seed from the TAGI repository was used for comparison purposes.

Source

https://github.com/CivML-PolyMtl/TAGI/blob/master/ToyExample/ToyExample_1D.m

References

Hernández-Lobato, J. M., & Adams, R. "Probabilistic backpropagation for scalable learning of bayesian neural networks." International Conference on Machine Learning. 2015.

ToyExample.y_val *Responses used in validation part for 1D toy problem*

Description

The orignal dataset represents a 1D regression problem from Hernández-Lobato & Adams (2015): $y = x^3 + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 9)$ and $x \in [-4, 4]$. In this dataset, x and y are normalized.

Usage

```
ToyExample.y_val
```

Format

A data frame with 20 rows and 1 variable y

Details

The dataset generated with the seed from the TAGI repository was used for comparison purposes.

Source

https://github.com/CivML-PolyMtl/TAGI/blob/master/ToyExample/ToyExample_1D.m

References

Hernández-Lobato, J. M., & Adams, R. "Probabilistic backpropagation for scalable learning of bayesian neural networks." International Conference on Machine Learning. 2015.

Index