Package 'BPHO'

July 31, 2009

Version 1.3-0

Title Bayesian Prediction with High-order Interactions

Author Longhai Li <longhai@math.usask.ca>

Maintainer Longhai Li <longhai@math.usask.ca>

Depends R (>= 2.5.1)

Description This software can be used in two situations. The first is to predict the next outcome based on the previous states of a discrete sequence. The second is to classify a discrete response based on a number of discreate covariates. In both situations, we use Bayesian logistic regression models that consider the high-order interactions. The models are trained with slice sampling method, a variant of Markov chain Monte Carlo. The time arising from using high-order interactions is reduced greatly by our compression technique that represents a group of original parameters as a single one in MCMC step.

License GPL (>=2)

URL \url{http://www.r-project.org}, \url{http://math.usask.ca/~longhai}

R topics documented:

comp_train_pred *User-level functions for compressing parameters, training the models with MCMC, and making predictions for test cases*

Description

The function comp_train_pred can be used for all of three tasks: compressing parameter, training the models with MCMC, and making prediction for test cases. When **new_comp=1**, it compresses parameters based on training cases and the information about parameter compression is written to the binary file ptn_file. When **new_comp=0**, it uses the existing ptn_file. When **iters_mc > 0**, it trains the models with Markov chain Monte Carlo and the Markov chain iterations are written to the binary file mc_file. The methods of writing to and reading from the files ptn_file and mc_file can be found from the documentations compression and training. When **iters_pred > 0** and **test_x** isn't empty, it predicts the responses of test cases and the result is written to the file pred_file and also returned as a value of this function.

The function cv_comp_train_pred is a short-cut function for performing cross-validation with the function comp_train_pred.

The argument is_sequence=1 indicates that a sequence prediction model is fitted to the data, and is_sequence=1 indicates that a general classification model based on discrete predictor variables is fitted.

Usage

```
comp_train_pred(
    ################## specify data information  ######################
    test_x = c(),train_x,train_y,no_cls,nos_features=c(),
    ################## specify for compression ########################
    is_sequence=1,order,ptn_file=c(), new_comp = 1, comp = 1,
    no_cases_ign = 0,
    ################## specify for priors  ############################
    alpha=1,log_sigma_widths=c(),log_sigma_modes=c(),
    ################## specify for mc sampling ########################
    mc_file=c(),start_over=TRUE,iters_mc=200,iters_bt=10,
    iters_sgm=50,w_bt=5,m_bt=20,w_sgm=1,m_sgm=20,ini_log_sigmas=c(),
    ################## specify for prediction #########################
    pred_file=c(),iter_b = 100,forward = 2,samplesize = 50)

cv_comp_train_pred(
    ###################### Specify data,order,no_fold #################
    no_fold=10,train_x,train_y,no_cls=c(),nos_features=c(),
    ###################### specify for compressing ####################
    is_sequence=1,order,ptn_file=c(),comp=1,no_cases_ign = 0,
    ###################### specify for priors  ########################
    alpha=1,log_sigma_widths=c(),log_sigma_modes=c(),
    ###################### specify for mc sampling ####################
```

```
mc_file=c(),iters_mc=200,iters_bt=10,iters_sgm=50,
w_bt=5,w_sgm=1,m_bt=20,m_sgm=20,ini_log_sigmas=c(),
###################### specify for prediction ######################
pred_file=c(),iter_b=100,forward=2,samplesize=50)
```

Arguments

test_x	Discrete features (also called inputs,covariates,independent variables, explanatory variables, predictor variables) of test data on which the predictions are based. The row is subject and the columns are inputs, which are coded with 1,2,..., with 0 reserved to represent that this input is not considered in a pattern. When the sequence prediction models are fitted, it is assumed that the first column is the state closest to the response. For example, a sequence 'x1,x2,x3,x4' is saved in test_x as 'x4,x3,x2,x1', for predicting the response 'x5'. It could be empty if there isn't prediction needed.
train_x	Discrete features of training data of the same format as test_x.
train_y	Discrete response of training data, a vector with length equal to the row of train_x. Assumed to be coded with 1,2,... no_cls .
no_cls	the number of possibilities (classes) of the response, default to the maximum value in train_y.
nos_features	a vector, with each element storing the number of possibilities (classes) of each feature, default to the maximum value of each feature.
is_sequence	is_sequence=1 indicates that sequence prediction models are fitted to the data, and is_sequence=0 indicates that general classification models based on discrete predictor variables are fitted.
no_fold	Number of folders in cross-validation.
order	the order of interactions considered, default to the total number of features, i.e. ncol(train_x).
ptn_file	a character string, the name of the binary file to which the compression result is written. The method of writing to and reading from ptn_file can be found from the documentation for compression.
new_comp	new_comp=1 indicates removing the old file ptn_file if it exists and doing the compression once again. new_comp=0 indicates using the old file ptn_file without doing compression once again. Note that when new_comp=0, the specification related to training cases does not take effect.
no_cases_ign	When the number of training cases for a pattern is no more than no_cases_ign, this pattern will be ignored, default to 0.
comp	comp=1 indicates doing compression, and comp=0 indicates using original parametrization. This is used only to make comparison. In practice, we definitely recommend using our compression technique to reduce the number of parameters.
alpha	alpha=1 indicates that Cauchy prior is used, alpha=2 indicates that Gaussian prior is used.

`log_sigma_widths, log_sigma_modes`
: two vectors of length `order+1`, which are interpreted as follows: the Gaussian distribution with location `log_sigma_modes[o]` and standard deviation `log_sigma_widths[o]` is the prior for 'log(sigmas[o])', which is the hyperparameter (width parameter of Gaussian distribution or Cauchy distribution) for the regression coefficients (i.e. 'beta's) associated with the interactions of order 'o'. If they are set to empty, the program will specify them automatically. By default, `log_sigma_widths` are all equal to 1, and `log_sigma_modes` starts at 2 for order 1 if Gaussian prior is used, and -1 if Cauchy prior is used, then decreases by 0.5 per order.

`mc_file`
: A character string, the name of the binary file to which Markov chain is written. The method of writing to and reading from `mc_file` can be found from the documentation for `training`.

`start_over`
: `start_over=TRUE` indicates that the existing file `mc_file` is deleted before a Markov chain sampling starts, otherwise the Markov chain will continue from the last iteration stored in `mc_file`.

`iters_mc,iters_bt,iters_sgm`
: `iters_mc` iterations of super-transition will be run. Each super-transition consists of `iters_bt` iterations of updating 'beta's, and for each updating of 'beta's, the hyperparameters 'log(sigma)'s are updated `iters_sgm` times. When `iters_mc=0`, no Markov chain sampling will be run and other arguments related to Markov chain sampling take no effect.

`w_bt, m_bt, w_sgm, m_sgm`
: `w_bt` is the amount of stepping-out in updating 'beta' with slice sampling, `m_bt` is the maximum number of stepping-out in slice sampling for updating 'beta'. `w_sgm` and `m_sgm` are intepreted similarly for sampling for 'log(sigma)'.

`ini_log_sigmas`
: Initial values of 'log(sigma)', default to `log_sigma_modes`.

`pred_file`
: A character string, the name of the file to which the prediction result is written. If `pred_file=c()`, the prediction result is printed out on screen (or sent to standard output).

`iter_b, forward, samplesize`
: Starting from `iter_b`, one of every `forward` Markov chain samples, with the number of total samples being <= `samplesize` and the maximum usable in the file `mc_file`, is used to make prediction.

Value

The function `comp_train_pred` returns the following values:

`times`
: The time in second for, as this order, compressing parameters, training the model, predicting for test cases

`pred_result`
: a data frame with first `no_cls` columns being the predictive probability and the next column being the predicted response value is returned.

`files`
: Three character strings: the 1st is the name of the file storing compression information, the 2nd is the name of the file storing Markov chain, and the 3rd one is the name of the file containing the detailed prediction result, i.e., `pred_result`.

The function cv_comp_train_pred returns the following additional values:

eval_details	a data frame. The first column is the true response, the second is the guessed value by taking the label of class with largest predictive probability, the third is indicator whether a wrong decision is make, the last column is the predictive probability at the true class.
error_rate	the proportion of wrong prediction.
amll	the average of minus log probabilities at true class, i.e. the average of the logarithms of the last column of eval_details.

Author(s)

Longhai Li, http://math.usask.ca/~longhai

References

http://math.usask.ca/~longhai/doc/seqpred/seqpred.abstract.html

See Also

gendata,compression,training,prediction

Examples

```
#######################################################################
########  These are demonstrations of using the whole package
#######################################################################

#######################################################################
########  Apply to Sequence Prediction Models
#######################################################################

## generate data from a hidden Markov model
data_hmm <- gen_hmm(n = 200, p = 10, no_h = 4, no_o = 2,
                    prob_h_stay = 0.8, prob_o_stay = 0.8)

## compress parameters (transforming features in training data.)
compress (features = data_hmm$X[-(1:100),], is_sequence = 1, order = 4,
          ptn_file = ".ptn_file.log")

## print summary information of '.ptn_file.log'
display_ptn (ptn_file = ".ptn_file.log")

## draw 50 samples with slice sampling, saved in '.mc_file.log'
training (train_y = data_hmm$y[-(1:100)], no_cls = 2,
          mc_file = ".mc_file.log", ptn_file = ".ptn_file.log",
          iters_mc = 50)

## draw 50 more samples from the last iteration in '.mc_file.log'
training (train_y = data_hmm$y[-(1:100)], no_cls = 2,
          mc_file = ".mc_file.log", ptn_file = ".ptn_file.log",
          iters_mc = 50, start_over = FALSE)
```

```
## display general information of Markov chain stored in '.mc_file.log'
summary_mc (mc_file = ".mc_file.log")

## find medians of Markov chain samples of all betas
med_betas <- medians_betas (mc_file = ".mc_file.log",
             iter_b = 50, forward = 1, samplesize = 50)

## take the id of group with highest absolute median
g_impt <- med_betas [1, "groupid"]

## particularly read `betas' by specifying the group and class id
plot ( read_betas (ix_g = g_impt, ix_cls = 2, mc_file = ".mc_file.log",
       iter_b = 50, forward = 1, samplesize = 50) )

## get information about pattern groups
display_ptn (".ptn_file.log", ix_g = g_impt )

## read Markov chain values of log-likelihood from  '.mc_file.log'
plot ( read_mc (group = "lprobs", ix = 1, mc_file = ".mc_file.log",
       iter_b = 50, forward = 1, samplesize = 50) )

## predict for test cases
pred_probs <- predict_bpho (test_x = data_hmm$X[1:100,],
              mc_file = ".mc_file.log", ptn_file = ".ptn_file.log",
              iter_b = 50, forward = 1, samplesize = 50 )

## evaluate predictions with true value of response
evaluate_prediction (data_hmm$y[1:100], pred_probs)

#####################################################################
########  Apply to General Classification Models
#####################################################################

## generating a classification data
data_class <- gen_bin_ho(n = 200, p = 3, order = 3, alpha = 1,
              sigmas = c(0.3,0.2,0.1), nos_features = rep(4,3), beta0 = 0)

## compress parameters (transforming features in training data.)
compress (features = data_class$X[-(1:100),], is_sequence = 0, order = 20,
          ptn_file = ".ptn_file.log")

## print summary information of '.ptn_file.log'
display_ptn (ptn_file = ".ptn_file.log")

## draw 50 samples with slice sampling, saved in '.mc_file.log'
training (train_y = data_class$y[-(1:100)], no_cls = 2,
          mc_file = ".mc_file.log", ptn_file = ".ptn_file.log",
          iters_mc = 50)

## draw 50 more samples from the last iteration in '.mc_file.log'
training (train_y = data_class$y[-(1:100)], no_cls = 2,
          mc_file = ".mc_file.log", ptn_file = ".ptn_file.log",
```

```
          iters_mc = 50, start_over = FALSE)

## display general information of Markov chain stored in '.mc_file.log'
summary_mc (mc_file = ".mc_file.log")

## find medians of Markov chain samples of all betas
med_betas <- medians_betas (mc_file = ".mc_file.log",
             iter_b = 50, forward = 1, samplesize = 50)

## take the id of group with highest absolute median
g_impt <- med_betas [1, "groupid"]

## particularly read `betas' by specifying the group and class id
plot ( read_betas (ix_g = g_impt, ix_cls = 2, mc_file = ".mc_file.log",
       iter_b = 50, forward = 1, samplesize = 50) )

## get information about pattern groups
display_ptn (".ptn_file.log", ix_g = g_impt )

## read Markov chain values of log-likelihood from  '.mc_file.log'
plot ( read_mc (group = "lprobs", ix = 1, mc_file = ".mc_file.log",
       iter_b = 50, forward = 1, samplesize = 50) )

## predict for test cases
pred_probs <- predict_bpho (test_x = data_class$X[1:100,],
              mc_file = ".mc_file.log", ptn_file = ".ptn_file.log",
              iter_b = 50, forward = 1, samplesize = 50 )

## evaluate predictions with true value of response
evaluate_prediction (data_class$y[1:100], pred_probs)

#######################################################################
########  Demonstrations of using a single function 'comp_train_pred'
#######################################################################

## generating a classification data
data_class <- gen_bin_ho(n = 200, p = 3, order = 3, alpha = 1,
              sigmas = c(0.3,0.2,0.1), nos_features = rep(4,3), beta0 = 0)

## carry out compression, training, and prediction with a function
comp_train_pred (
    ################## specify data information  ######################
    test_x = data_class$X[1:100,], train_x = data_class$X[-(1:100),],
    train_y = data_class$y[1:100], no_cls = 2, nos_features = rep(4,3),
    ################## specify for compression ########################
    is_sequence = 0, order = 3, ptn_file=".ptn.log", new_comp = 1, comp = 1,
    ################## specify for priors  ############################
    alpha = 1, log_sigma_widths = c(), log_sigma_modes = c(),
    ################## specify for mc sampling ########################
    mc_file = ".mc.log", iters_mc = 200, iters_bt = 10,
    iters_sgm = 50, w_bt = 5, m_bt = 20, w_sgm = 1, m_sgm = 20,
    ini_log_sigmas = c(), start_over = TRUE,
    ################## specify for prediction #########################
```

```
    pred_file = "pred_file.csv", iter_b = 100, forward = 2, samplesize = 50)

## evaluate predictions with true value of response
evaluate_prediction (data_class$y[1:100], read.csv("pred_file.csv") )

#####################################################################
#####  Demonstrations of using a single function 'cv_comp_train_pred'
#####################################################################

## generating a classification data
data_class <- gen_bin_ho(n = 200, p = 3, order = 3, alpha = 1,
              sigmas = c(0.3,0.2,0.1), nos_features = rep(4,3), beta0 = 0)

## carry out cross-validation with data set data_class
cv_comp_train_pred (
    ################## Specify data,order,no_fold ####################
    no_fold = 2, train_x = data_class$X,train_y = data_class$y,
    no_cls = 2, nos_features = rep(4,3),
    ################## specify for compression ########################
    is_sequence = 0, order = 3, ptn_file=".ptn.log",  comp = 1,
    ################## specify for priors  ############################
    alpha = 1,log_sigma_widths = c(),log_sigma_modes = c(),
    ################## specify for mc sampling ########################
    mc_file = ".mc.log", iters_mc = 200, iters_bt = 10,
    iters_sgm = 50,w_bt = 5,m_bt = 20,w_sgm = 1,m_sgm = 20,
    ini_log_sigmas = c(),
    ################## specify for prediction #########################
    pred_file = "pred_file.csv", iter_b = 100, forward = 2, samplesize = 50)
```

`compression` *Functions related to parameter compression*

Description

The function `compress` groups the patterns in a way such that the interaction patterns in a group are expressed by the same training cases. In training the models with MCMC, we need to use only one parameter for each group, which represents the sum of all the parameters in this group. The original parameters are seemly compressed. A large amount of training time is saved by this compression techniques.

The result of this grouping is saved in a binary file in a way such that it can be retrieved as a linked list in C, with each node consisting of a description (an integer vector of fixed length) of the group of patterns and the indice (an integer vector of varying length, with 0 for the first training case) of training cases expressing this group of patterns. This file is needed to train the models with MCMC and to predict the responses of test cases using the function `comp_train_pred`.

The function `display_ptn` displays the summary information about this compression, such as the number of groups and total number of patterns expressed by the training cases. When `ix_g` is nonempty, it also displays the detailed information about the groups specified by `ix_g`, such as the pattern description and the indice of training cases associated with this group.

Usage

```
compress (
    features, is_sequence=1, order=ncol(features), ptn_file,
    nos_features=c(), comp=1, quiet=1, no_cases_ign=0 )
display_ptn ( ptn_file, ix_g=c() )
```

Arguments

features	Discrete features (also called features,covariates,independent variables, explanatory variables, predictor variables) of training data on which the predictions are based. The row is subject and the columns are inputs, which are coded with 1,2,..., with 0 reserved to represent that this input is not considered in a pattern. When the sequence prediction models are fitted, it is assumed that the first column is the state closest to the response. For example, a sequence 'x1,x2,x3,x4' is saved in test_x as 'x4,x3,x2,x1', for predicting the response 'x5'.
is_sequence	is_sequence=1 indicates that sequence prediction models are fitted to the data, and is_sequence=0 indicates that general classification models based on discrete predictor variables are fitted.
order	the order of interactions considered, default to total number of features, i.e. ncol(features).
ptn_file	a character string, the name of the binary file to which the compression result is written.
nos_features	a vector, with each element storing the number of possibilities (classes) of each feature, default to the maximum value of each feature.
comp	comp=1 indicates doing compression, and comp=0 indicates using original parametrization. This is used only to make comparison. In practice, I of course recommend using our compression technique to reduce the number of parameters.
quiet	If quiet=0, some messages during compression are printed on screen for monitor the compression, if quiet=1 the function works silently.
no_cases_ign	When the number of training cases for a pattern is no more than no_cases_ign, this pattern will be ignored, default to 0.
ix_g	an integer vector, containing the indice of groups whose information you want to display, with 1 for the first group.

Value

The function compress returns no value. Instead, it saves the result of compression in the file ptn_file.

The function display_ptn returns a vector of 6 numbers. Their meanings are as follows: is.sequence – indictor whether a sequence model is fitted,order – the maximum order of interactions considered, #groups – the number of groups found, #patterns – the number of interaction patterns expressed by the training cases,#cases – the number of training cases, #features – the number of features.

When ix_g is nonempty, it also displays the details about the queried groups. The information printed on screen for each group is read as follows. Under **superpatterns**, it displays a compact description of the pattern group, which is in a special format defined in the references associated with this software. Under **expression**, it displays the indice of training cases that express this group of patterns. Under **sigmas**, it displays the number of patterns with a certain order, starting from order 0. This information is needed to compute the width parameter of the regression coeficient associated with this group from the values of hyperparameters 'sigma's.

See Also

comp_train_pred, training, prediction

Examples

```
## generate features
features <- gen_X(50,5,2)

## compressing the parameter based on 'features'
compress ( features, is_sequence=1, ptn_file=".ptn_file.log")

## display the summary information in the file ".ptn_file.log"
display_ptn(".ptn_file.log")

## display the information for group #2 and #3
display_ptn(".ptn_file.log",ix_g=c(2,3))
```

---

training *Functions related to Markov chain sampling*

Description

The models are trained with Markov chain Monte Carlo (MCMC) methods. Slice sampling is used to update 'beta's, the regression coefficients for groups, and 'log(sigma)', where 'sigma' is the width parameter of the prior for 'beta'.

The function training carries out the Markov chain sampling, saving the Markov chain samples in a binary file mc_file.

The function summary_mc displays the summary information in the file mc_file.

The function read_mc reads the Markov chain samples from the file mc_file at given iterations.

The function read_betas is based on the function read_mc. It specifically reads the 'beta' for given group and class identities.

The function medians_betas returns the medians of the Markov chain samples for all 'beta's at specified iterations. This function is for discovering important interaction patterns. An interaction pattern with large absolute medians is suspected to be an important pattern for predicting the response.

Usage

```
training (
    ################## specify for data  ##############################
    train_y, no_cls, mc_file, ptn_file,
    ################## specify for priors  ############################
    alpha=1, log_sigma_modes=c(), log_sigma_widths=c(),
    ################## specify for slice sampling #####################
    iters_mc=200, iters_bt=10,  iters_sgm=50,
    w_bt=5, m_bt=20, w_sgm=1, m_sgm=20, ini_log_sigmas=c(),
    start_over=TRUE )
summary_mc(mc_file)
read_mc(group,ix, mc_file, iter_b=1,forward=1,samplesize=c(),quiet=1)
read_betas(ix_g, ix_cls, mc_file, iter_b=1,forward=1,samplesize=c(),quiet=1)
medians_betas(mc_file, iter_b=1, forward=1, samplesize=c() )
```

Arguments

`mc_file`	A character string, the name of the binary file to which Markov chain is written.
`group`	A character string giving the group name of values. It can be one of 'lprobs',lsigmas','betas', 'evals'. Group 'lprobs' contains: the values of log probabilities of data given the values of 'beta's (identified by `ix=0`), the value of log prior of 'beta's given 'sigma's (identified by `ix=1`), the value of log prior of 'log(sigma)'s (identified by `ix=2`), and the value of log posterior (identified by `ix=3`), which is the sum of the previous three values. Group 'lsigmas' contains: the values of hyperparameters 'log(sigma)', with `ix` indicating the order, starting from 0. Group 'betas' contains: the values of 'betas', with `ix` indicating the index of 'beta'. The 'beta's in each iteration is placed as that the `no_cls` values of 'beta's for pattern group 'i' are followed by the next `no_cls` values for pattern group 'i+1'. The smallest index is 0. Group 'evals' contains: the average times of evaluating the posterior distribution in updating each 'beta' using slice sampling (identified by `ix=0`), and the average rejection rate of updating each 'log(sigma)' with Metropolis sampling (identified by `ix=1`).
`ix`	index of parameters inside each group, as discussed for `group` above.
`ix_g`	index of pattern group, starting from 1.
`ix_cls`	index of class, ranging from **1** to **`no_cls`**.
`iter_b, forward, samplesize`	Starting from `iter_b`, one of every `forward` Markov chain samples, with the number of total samples being <= `samplesize` and the maximum usable in the file `mc_file`, is read.
`train_y`	Discrete response of training data. Assumed to be coded with 1,2,... `no_cls`.
`no_cls`	the number of possibilities (classes) of the response, default to the maximum value in `train_y`.

`alpha` `alpha=1` indicates that Cauchy prior is used, `alpha=2` indicates that Gaussian prior is used.

`log_sigma_widths, log_sigma_modes`
two vectors of length `order+1`, which are interpreted as follows: the Gaussian distribution with location `log_sigma_modes[o]` and standard deviation `log_sigma_widths[o]` is the prior for 'log(sigmas[o])', which is the hyperparameter (width parameter of Gaussian distribution or Cauchy distribution) for the regression coefficients (i.e. 'beta's) associated with the interactions of order 'o'. If they are set to empty, the program will specify them automatically. By default, `log_sigma_widths` are all equal to 1, and `log_sigma_modes` starts at 2 for order 1 if Gaussian prior is used, and -1 if Cauchy prior is used, then decreases by 0.5 per order.

`ptn_file` a character string, the name of the binary file where the compression result is saved. The method of writing to and reading from `ptn_file` can be found from the documentation for `compression`.

`iters_mc,iters_bt,iters_sgm`
`iters_mc` iterations of super-transition will be run. Each super-transition consists of `iters_bt` iterations of updating 'beta's, and for each updating of 'beta's, the hyperparameters 'log(sigma)'s are updated `iters_sgm` times. When `iters_mc=0`, no Markov chain sampling will be run and other arguments related to Markov chain sampling take no effect.

`w_bt, m_bt, w_sgm, m_sgm`
`w_bt` is the amount of stepping-out in updating 'beta' with slice sampling, `m_bt` is the maximum number of stepping-out in slice sampling for updating 'beta'. `w_sgm` and `m_sgm` are intepreted similarly for sampling for 'log(sigma)'.

`ini_log_sigmas`
Initial values of 'log(sigma)', default to `log_sigma_mode`.

`quiet` `quiet=1` suppresses the messages printed during reading the file `mc_file`.

`start_over` `start_over=TRUE` indicates that the existing file `mc_file` is deleted before a Markov chain sampling starts, otherwise the Markov chain will continue from the last iteration stored in `mc_file`.

Value

The function `summary_mc` returns a vector with names as `#iters,#class,#groups,order,alpha`.

The function `read_mc` returns the Markov chain samples for a variable at specified iterations.

The function `read_betas` returns the Markov chain samples for a 'beta' at specified iterations.

The function `medians_betas` returns the medians of Markov chain samples of all 'beta's at given iterations, displayed as a matrix, with rows for different pattern group, and columns for different value of response.

The function `training` returns no value. Instead, the Markov chain samples are written to the binary file `mc_file`.

See Also

comp_train_pred,compression,prediction

Examples

```
## these functions are demonstrated in the section `comp_train_pred'.
```

---

prediction *Functions related to prediction*

Description

The function predict_bpho predicts the response of test cases.

The function evaluate_prediction evaluates the performance of the prediction in terms of average minus log probabilities and error rate. The function split_cauchy draws samples from a Cauchy distribution of two variables constraint to that their sum is fixed.

Usage

```
predict_bpho(test_x,mc_file,ptn_file,iter_b,forward,samplesize)
evaluate_prediction(test_y,pred_result,file_eval_details=c())
split_cauchy(n,s, sigma1,sigmasum,debug=1)
```

Arguments

test_x	Discrete features (also called inputs,covariates,independent variables, explanatory variables, predictor variables) of test data on which the predictions are based. The row is subject and the columns are inputs, which are coded with 1,2,..., with 0 reserved to represent that this input is not considered in a pattern. When the sequence prediction models are fitted, it is assumed that the first column is the state closest to the response. For example, a sequence 'x1,x2,x3,x4' is saved in test_x as 'x4,x3,x2,x1', for predicting the response 'x5'.
test_y	Discrete responses of test data, a vector with length equal to the row of test_x. Assumed to be coded with 1,2,... no_cls.
ptn_file	a character string, the name of the binary file to which the compression result is saved. The method of writing to and reading from ptn_file can be found from the documentation compression.
mc_file	A character string, the name of the binary file to which Markov chain is written. The method of writing to and reading from mc_file can be found from the documentation training.
iter_b, forward, samplesize	Starting from iter_b, one of every forward Markov chain samples, with the number of total samples being <= samplesize and the maximum usable in the file mc_file, is used to make prediction.

`pred_result` the value returned from the function `predict_bpho`.

`file_eval_details`
the details of evaluation is sent to the file `file_eval_details`.

`n` number of samples one wishes to obtain.

`s` sum of two Cauchy random variables.

`sigma1` scale parameter for the first Cauchy random variable.

`sigmasum` the sum of scale parameters for two Cauchy random variables.

`debug` indicator whether you are debugging the C program.

Value

The function `predict_bpho` returns a data frame, with the first `no_cls` columns storing the predictive probabilities for each class, and the last column is the guess for the response by choosing the label of the class with largest predictive probability.

The function `evaluate_prediction` returns the following values:

`eval_details` a data frame. The first column is the true response, the second is the guessed value by taking the label of class with largest predictive probability, the third is indicator whether a wrong decision is make, the last column is the predictive probability at the true class.

`error_rate` the proportion of wrong prediction.

`amll` the average of minus log probabilities at true class, i.e. the average of the logarithms of the last column of `eval_details`.

The function `split_cauchy` returns a vector of `n` random numbers.

See Also

`comp_train_pred`,`compression`,`training`

Examples

```
## the function `predict_bpho' is demonstrated with the function
## `comp_train_pred' which calls `predict_bpho' inside.

## examples of 'evaluate_prediction' can be found from
## the documentation for comp_train_pred.

## testing the function split_cauchy
split_cauchy(100,10,1,5)
```

gendata *Functions for generating data sets*

Description

gen_hmm generates sequences using hidden Markov models. gen_bin_ho generates general discrete data using logistic models, with high-order interactions considered; the response is binary. text_to_3number converts an English text file into sequence of 1 (special symbols such as space, symbol),2 (vowl),3 (consonant). text_to_number converts an English text into sequence of 1 - 27, 1-26 for letter a-z, and 27 for all other symbols.

Usage

```
gen_hmm(n,p,no_h,no_o,prob_h_stay, prob_o_stay)
gen_bin_ho(n,p,order,alpha,sigmas,nos_features,beta0)
text_to_number(p,file)
text_to_3number(p,file)
gen_X(n,p,K)
```

Arguments

n	number of cases.
p	number of features, or length of sequence.
K	number of possibilities for each feature.
no_h	number of states of hidden Markov chain.
no_o	number of states of output in hidden Markov model.
prob_h_stay	In simulating the hidden Markov chain, a chain will stay in its previous state with probability prob_h_stay, and move to other states with some minor probabilities adding up to 1-prob_h_stay.
prob_o_stay	In simulating the output state of hidden Markov model, the "output" is equal to ("hidden state" mod no_o)+1 with probability prob_o_stay and equally likely other states.
order	the order of interactions considered in simulating data from general classification models.
alpha	alpha=2 indicates that Gaussian distributions are used to generate the “beta"s and alpha=1 indicates that Cauchy distributions are used.
sigmas	hyperparameters in generating "beta"s, a vector of length order.
nos_features	number of states for each feature, i.e., the number of possibilities for each feature. A vector of length p.
beta0	intercept of linear function in generating classification data.
file	name of the file containing text file, a character string.

Value

X	values of predictors, a matrix. Each row is a case. For sequence, the data for each case (a row) is placed in the reverse order of time. For example, sequence "x1,x2,x3" is represented with a row of X: x3,x2,x1. The values of predictor X are coded by 1,2,3,...,nos_features. The function gen_X generates only this matrix.
y	values of the response, a vector, coded by 1,2,...
betas	a matrix of two columns saving the values of "betas" used in generating classification data. The first column is the absolute identity of this beta, and the 2nd column is the value. The total number of "betas" is saved in no_betas.

See Also

comp_train_pred

Examples

```
data_hmm <- gen_hmm(100,10,8,2,0.8,0.8)
data_bin_ho <- gen_bin_ho(100,3,2,1,c(5,2),c(3,3,3),0)
X <- gen_X(100,5,3)
```

Index