Package ‘VBphenoR’

September 15, 2025

Title Variational Bayes for Latent Patient Phenotypes in EHR

Version 1.0.0

Description

Identification of Latent Patient Phenotype from Electronic Health Records (EHR) Data using Variational Bayes Gaussian Mixture Model for Latent Class Analysis and Variational Bayes regression for Biomarker level shifts, both implemented by Coordinate Ascent Variational Inference algorithms. Variational methods are used to enable Bayesian analysis of very large Electronic Health Records data. For VB GMM details see Bishop (2006,ISBN:9780-387-31073-2). For Logistic VB see Jaakkola and Jordan (2000) <doi:10.1023/A:1008932416310>.

License MIT + file LICENCE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests rmarkdown, testthat (>= 3.0.0)

Imports stats, CholWishart, pracma, knitr, utils, dbscan, data.table, ggplot2

Depends R (>= 3.5.0)

LazyData true

Config/testthat/edition 3

URL `https://github.com/buckleybrian/VBphenoR`, `https://buckleybrian.github.io/VBphenoR/`

BugReports `https://github.com/buckleybrian/VBphenoR/issues`

Config/Needs/website rmarkdown

NeedsCompilation no

Author Brian Buckley [aut, cre, cph] (ORCID: <`https://orcid.org/0000-0002-5478-5830`>),
Adrian O'Hagan [aut] (Co-author),
Marie Galligan [aut] (Co-author)

Maintainer Brian Buckley <`brian.buckley.1@ucdconnect.ie`>

Repository CRAN

Date/Publication 2025-09-15 07:40:08 UTC

Contents

`logit_CAVI` *Variational inference for Bayesian logistic regression using CAVI algorithm*

Description

Variational inference for Bayesian logistic regression using CAVI algorithm

Usage

```
logit_CAVI(
  X,
  y,
  prior,
  delta = 1e-16,
  maxiters = 10000,
  verbose = FALSE,
  progressbar = TRUE
)
```

Arguments

`X` The input design matrix. Note the intercept column vector is assumed included.

`y` The binary response.

`prior` Prior for the logistic parameters.

`delta` The ELBO difference tolerance for conversion.

`maxiters` The maximum iterations to run if convergence is not achieved.

`verbose` A diagnostics flag (off by default).

`progressbar` A visual progress bar to indicate iterations (on by default).

Value

A list containing:

- error - An error message if convergence failed or the number of iterations to achieve convergence.
- mu - A vector of posterior means.
- Sigma - A vector of posterior variances.
- Convergence - A vector of the ELBO at each iteration.
- LBDifference - A vector of ELBO differences between each iteration.
- xi - A vector of log-odds per X row.

Examples

```
# Use Old Faithful data to show the effect of VB GMM Priors,
# stopping on ELBO reverse parameter or delta threshold
# -------------------------------------------------------------------------------

gen_path <- tempdir()
data("faithful")
X <- faithful
P <- ncol(X)

# Run with 4 presumed components for demonstration purposes
k = 4

# -------------------------------------------------------------------------------
# Plotting
# -------------------------------------------------------------------------------

#' Plots the GMM components with centroids
#'
#' @param i List index to place the plot
#' @param gmm_result Results from the VB GMM run
#' @param var_name Variable to hold the GMM hyperparameter name
#' @param grid Grid element used in the plot file name
#' @param fig_path Path to the directory where the plots should be stored
#'
#' @returns
#' @importFrom ggplot2 ggplot
#' @importFrom ggplot2 aes
#' @importFrom ggplot2 geom_point
#' @importFrom ggplot2 scale_color_discrete
#' @importFrom ggplot2 stat_ellipse
#' @export

do_plots <- function(i, gmm_result, var_name, grid, fig_path) {
  dd <- as.data.frame(cbind(X, cluster = gmm_result$z_post))
  dd$cluster <- as.factor(dd$cluster)
  # The group means
  # ---------------------------------------------------------------------------
  mu <- as.data.frame( t(gmm_result$q_post$m) )

  # Plot the posterior mixture groups
  # ---------------------------------------------------------------------------
  cols <- c("#1170AA", "#55AD89", "#EF6F6A", "#D3A333", "#5FEFE8", "#11F444")
  p <- ggplot2::ggplot() +
  ggplot2::geom_point(dd, mapping=ggplot2::aes(x=eruptions, y=waiting, color=cluster)) +
    ggplot2::scale_color_discrete(cols, guide = 'none') +
  ggplot2::geom_point(mu, mapping=ggplot2::aes(x = eruptions, y = waiting), color="black",
               pch=7, size=2) +
    ggplot2::stat_ellipse(dd, geom="polygon",
                 mapping=ggplot2::aes(x=eruptions, y=waiting, fill=cluster),
                 alpha=0.25)

  grids <- paste((grid[,i]), collapse = "_")
  ggplot2::ggsave(filename=paste0(var_name,"_",grids,".png"), plot=p, path=fig_path,
         width=12, height=12, units="cm", dpi=600, create.dir = TRUE)
}

# -----------------------------------------------------------------------------
# Dirichlet alpha
# -----------------------------------------------------------------------------
alpha_grid <- data.frame(c1=c(1,1,1,1),
                         c2=c(1,92,183,183),
                         c3=c(183,92,183,183))
init <- "kmeans"

for (i in 1:ncol(alpha_grid)) {
  prior <- list(
    alpha = as.integer(alpha_grid[,i]) # set most of the weight on one component
  )

  gmm_result <- vb_gmm_cavi(X=X, k=k, prior=prior, delta=1e-8, init=init,
                            verbose=FALSE, logDiagnostics=FALSE)
  do_plots(i, gmm_result, "alpha", alpha_grid, gen_path)
}

# -----------------------------------------------------------------------------
# Normal-Wishart beta for precision proportionality
# -----------------------------------------------------------------------------
beta_grid <- data.frame(c1=c(0.1,0.1,0.1,0.1),
                        c2=c(0.9,0.9,0.9,0.9))
init <- "kmeans"

for (i in 1:ncol(beta_grid)) {
  prior <- list(
    beta = as.numeric(beta_grid[,i])
  )

  gmm_result <- vb_gmm_cavi(X=X, k=k, prior=prior, delta=1e-8, init=init,
                            verbose=FALSE, logDiagnostics=FALSE)
  do_plots(i, gmm_result, "beta", beta_grid, gen_path)
}

# ------------------------------------------------------------------------------
# Normal-Wishart W0 (assuming variance matrix) & logW
# ------------------------------------------------------------------------------

w_grid <- data.frame(w1=c(0.001,0.001),
                     w2=c(2.001,2.001))
init <- "kmeans"

for (i in 1:nrow(w_grid)) {
  w0 = diag(w_grid[,i],P)
  prior <- list(
    W = w0,
    logW = -2*sum(log(diag(chol(w0))))
  )

  gmm_result <- vb_gmm_cavi(X=X, k=k, prior=prior, delta=1e-8, init=init,
                            verbose=FALSE, logDiagnostics=FALSE)
  do_plots(i, gmm_result, "w", w_grid, gen_path)
}
```

run_Model *Run the Variational Bayes patient phenotyping model*

Description

Run the Variational Bayes patient phenotyping model

Usage

```
run_Model(
  biomarkers,
  gmm_X,
  logit_X,
  gmm_delta = 1e-06,
  logit_delta = 1e-16,
  gmm_maxiters = 200,
  logit_maxiters = 10000,
  gmm_init = "kmeans",
  gmm_initParams = NULL,
  gmm_prior = NULL,
  logit_prior = NULL,
  gmm_stopIfELBOReverse = FALSE,
  gmm_verbose = FALSE,
  logit_verbose = FALSE,
  gmm_progressbar = FALSE,
  logit_progressbar = FALSE
)
```

Arguments

`biomarkers` The EHR variables that are biomarkers. This is a vector of data column names corresponding to the biomarker variables.

`gmm_X` n x p data matrix (or data frame that will be converted to a matrix).

`logit_X` The input design matrix. Note the intercept column vector is assumed included.

`gmm_delta` Change in ELBO that triggers algorithm stopping.

`logit_delta` Change in ELBO that triggers algorithm stopping.

`gmm_maxiters` The maximum iterations for VB GMM.

`logit_maxiters` The maximum iterations for VB logit.

`gmm_init` Initialize the clusters c("random", "kmeans", "dbscan").

`gmm_initParams` Parameters for an initialiser requiring its own parameters e.g. dbscan requires ’eps’ and ’minPts’.

`gmm_prior` An informative prior for the GMM.

`logit_prior` An informative prior for the logit.

`gmm_stopIfELBOReverse`
Stop the VB iterations if the ELBO reverses direction (TRUE or FALSE).

`gmm_verbose` Print out information per iteration to track progress in case of long-running experiments.

`logit_verbose` Print out information per iteration to track progress in case of long-running experiments.

`gmm_progressbar`
Show a progressbar driven by the GMM variational iterations.

`logit_progressbar`
Show a progressbar driven by the logit variational iterations.

Value

A list containing:

- prevalence - The mean probability of latent phenotype given the data and priors.
- biomarker_shift - A data frame containing the biomarker shifts from normal for the phenotype.
- gmm - The VB GMM results. For details see help(vb_gmm_cavi).
- logit - The VB Logit results. For details see help(logit_CAVI).

Examples

```
##Example 1: Use the internal Sickle Cell Disease data to find the rare
##           phenotype.  SCD is extremely rare so we use DBSCAN to initialise
##           the VB GMM. We also use an informative prior for the mixing
##           coefficient and stop iterations when the ELBO starts to reverse
##           so that we stop when the minor (SCD) component is reached.

library(data.table)

# Load the SCD example data supplied with the VBphenoR package
data(scd_cohort)

# We will use the SCD biomarkers to discover the SCD latent class.
# X1 is the data matrix for the VB GMM.
X1 <- scd_cohort[,.(CBC,RC)]

# We need to supply DBSCAN hyper-parameters as we will initialise VBphenoR
# with DBSCAN. See help(DBSCAN) for details of these parameters.
initParams <- c(0.15, 5)
names(initParams) <- c('eps','minPts')

# Set an informative prior for the VB GMM mixing coefficient alpha
# hyper-parameter
prior_gmm <- list(
  alpha = 0.001
)

# Set informative priors for the beta coefficients of the VB logit
prior_logit <- list(mu=c(1,
                   mean(scd_cohort$age),
                   mean(scd_cohort$highrisk),
                   mean(scd_cohort$CBC),
                   mean(scd_cohort$RC)),
              Sigma=diag(1,5))          # Simplest isotropic case

# X2 is the design matrix for the VB logit
X2 <- scd_cohort[,.(age,highrisk,CBC,RC)]
X2[,age:=as.numeric(age)]
X2[,highrisk:=as.numeric(highrisk)]
X2[,Intercept:=1]
setcolorder(X2, c("Intercept","age","highrisk","CBC","RC"))

# Run the patient phenotyping model

# Need to state what columns are the biomarkers
biomarkers <- c('CBC', 'RC')
set.seed(123)

pheno_result <- run_Model(biomarkers,
                        gmm_X=X1, gmm_init="dbscan",
                        gmm_initParams=initParams,
                        gmm_maxiters=20, gmm_prior=prior_gmm,
                        gmm_stopIfELBOReverse=TRUE,
                        logit_X=X2, logit_prior=prior_logit
)

# S3 print method
print(pheno_result)
```

`scd_cohort` *Synthetic Sickle Cell Anaemia data*

Description

This data is a transformed version of the SCD data from the paper by Al-Dhamari et al. Synthetic datasets for open software development in rare disease research. Orphanet J Rare Dis 19, 265 (2024).We have retained a subset of the data columns that are relevant to our model and transformed the data into a representative cohort by retaining an expected prevalence of SCD (0.3%), with the rest converted to non-SCD patients by distributing the biomarker values around a healthy value. These columns are described below.

Usage

```
scd_cohort
```

Format

`scd_cohort`**:**

A data frame with 100,403 rows and 9 columns:

age Patient Age

sex Patient gender assuming only Male and Female genders

race Patient race. One of "Others", "African-American", "European-American"

birthDate Patient birth date

diagDate Patient diagnosis date

CBC Complete Blood Count biomarker test in g/dL

RC Reticulocytes Count biomarker test in % Reticulocytes

highrisk Flag for high risk ethnicity

SCD Flag indicating SCD observations to test model performance

Source

Al-Dhamari (2024) doi:10.1186/s13023-024-03254-2.

`vb_gmm_cavi` *Main algorithm function for the VB CAVI GMM*

Description

Main algorithm function for the VB CAVI GMM

Usage

```
vb_gmm_cavi(
  X,
  k,
  prior = NULL,
  delta = 1e-06,
  maxiters = 5000,
  init = "kmeans",
  initParams = NULL,
  stopIfELBOReverse = FALSE,
  verbose = FALSE,
  logDiagnostics = FALSE,
  logFilename = "vb_gmm_log.txt",
  progressbar = TRUE
)
```

Arguments

`X` n x p data matrix (or data frame that will be converted to a matrix).

`k` guess for the number of mixture components.

`prior` Prior for the GMM parameters.

`delta` change in ELBO that triggers algorithm stopping.

`maxiters` maximum iterations to run if delta does not stop the algorithm already.

`init` initialize the clusters c("random", "kmeans", "dbscan").

`initParams` initialization parameters for dbscan. NULL if dbscan not selected for init.

`stopIfELBOReverse` stop the run if the ELBO at iteration t is detected to have reversed from iteration t-1.

`verbose` print out information per iteration to track progress in case of long-running experiments.

`logDiagnostics` log detailed diagnostics. If TRUE, a diagnostics RDS file will be created using the path specified in logFilename.

`logFilename` the filename of the diagnostics log.

`progressbar` A visual progress bar to indicate iterations (on by default).

Value

A list containing:

- error - An error message if convergence failed or the number of iterations to achieve convergence.
- z_post - A vector of posterior cluster mappings.
- q_post - A list of the fitted posterior Q family. q_post includes:
  - alpha - The Dirichlet prior for the mixing coefficient
  - beta - The proportionality of the precision for the Normal-Wishart prior
  - m - The mean vector for the Normal part of the Normal-Wishart prior
  - v - The degrees of freedom of the Wishart gamma ensuring it is well defined
  - U - The W hyperparameter of the Wishart conjugate prior for the precision of the mixtures. k sets of D x D symmetric, positive definite matrices.
  - logW - The logW term used to calculate the expectation of the mixture component precision. A vector of k.
  - R - The responsibilities. An n x k matrix.
  - logR - The log of responsibilities. An n x k matrix.
- elbo - A vector of the ELBO at each iteration.
- elbo_d - A vector of ELBO differences between each iteration.

Examples

```
# Use Old Faithful data to show the effect of VB GMM Priors,
# stopping on ELBO reverse parameter or delta threshold
# -----------------------------------------------------------------------------

gen_path <- tempdir()
data("faithful")
X <- faithful
P <- ncol(X)

# Run with 4 presumed components for demonstration purposes
k = 4

# -----------------------------------------------------------------------------
# Plotting
# -----------------------------------------------------------------------------

#' Plots the GMM components with centroids
#'
#' @param i List index to place the plot
#' @param gmm_result Results from the VB GMM run
#' @param var_name Variable to hold the GMM hyperparameter name
#' @param grid Grid element used in the plot file name
#' @param fig_path Path to the directory where the plots should be stored
#'
#' @returns
#' @importFrom ggplot2 ggplot
#' @importFrom ggplot2 aes
#' @importFrom ggplot2 geom_point
#' @importFrom ggplot2 scale_color_discrete
#' @importFrom ggplot2 stat_ellipse
#' @export

do_plots <- function(i, gmm_result, var_name, grid, fig_path) {
  dd <- as.data.frame(cbind(X, cluster = gmm_result$z_post))
  dd$cluster <- as.factor(dd$cluster)

  # The group means
  # ----------------------------------------------------------------------------
  mu <- as.data.frame( t(gmm_result$q_post$m) )

  # Plot the posterior mixture groups
  # ----------------------------------------------------------------------------
  cols <- c("#1170AA", "#55AD89", "#EF6F6A", "#D3A333", "#5FEFE8", "#11F444")
  p <- ggplot2::ggplot() +
  ggplot2::geom_point(dd, mapping=ggplot2::aes(x=eruptions, y=waiting, color=cluster)) +
    ggplot2::scale_color_discrete(cols, guide = 'none') +
  ggplot2::geom_point(mu, mapping=ggplot2::aes(x = eruptions, y = waiting), color="black",
               pch=7, size=2) +
    ggplot2::stat_ellipse(dd, geom="polygon",
                 mapping=ggplot2::aes(x=eruptions, y=waiting, fill=cluster),
                 alpha=0.25)

  grids <- paste((grid[,i]), collapse = "_")
  ggplot2::ggsave(filename=paste0(var_name,"_",grids,".png"), plot=p, path=fig_path,
        width=12, height=12, units="cm", dpi=600, create.dir = TRUE)
}

# -------------------------------------------------------------------------------
# Dirichlet alpha
# -------------------------------------------------------------------------------
alpha_grid <- data.frame(c1=c(1,1,1,1),
                         c2=c(1,92,183,183),
                         c3=c(183,92,183,183))
init <- "kmeans"

for (i in 1:ncol(alpha_grid)) {
  prior <- list(
    alpha = as.integer(alpha_grid[,i]) # set most of the weight on one component
  )

  gmm_result <- vb_gmm_cavi(X=X, k=k, prior=prior, delta=1e-8, init=init,
                            verbose=FALSE, logDiagnostics=FALSE)
  do_plots(i, gmm_result, "alpha", alpha_grid, gen_path)
}

# -------------------------------------------------------------------------------
# Normal-Wishart beta for precision proportionality
# -------------------------------------------------------------------------------
```

```
beta_grid <- data.frame(c1=c(0.1,0.1,0.1,0.1),
                        c2=c(0.9,0.9,0.9,0.9))
init <- "kmeans"

for (i in 1:ncol(beta_grid)) {
  prior <- list(
    beta = as.numeric(beta_grid[,i])
  )

  gmm_result <- vb_gmm_cavi(X=X, k=k, prior=prior, delta=1e-8, init=init,
                            verbose=FALSE, logDiagnostics=FALSE)
  do_plots(i, gmm_result, "beta", beta_grid, gen_path)
}

# --------------------------------------------------------------------------------
# Normal-Wishart W0 (assuming variance matrix) & logW
# --------------------------------------------------------------------------------

w_grid <- data.frame(w1=c(0.001,0.001),
                     w2=c(2.001,2.001))
init <- "kmeans"

for (i in 1:nrow(w_grid)) {
  w0 = diag(w_grid[,i],P)
  prior <- list(
    W = w0,
    logW = -2*sum(log(diag(chol(w0))))
  )

  gmm_result <- vb_gmm_cavi(X=X, k=k, prior=prior, delta=1e-8, init=init,
                            verbose=FALSE, logDiagnostics=FALSE)
  do_plots(i, gmm_result, "w", w_grid, gen_path)
}
```

VB_GMM_ELBO *Calculate the Evidence Lower Bound (ELBO)*

Description

Calculate the Evidence Lower Bound (ELBO)

Usage

```
VB_GMM_ELBO(X, p, n, q_post, prior)
```

Arguments

`X` n x p data matrix (or data frame that will be converted to a matrix).

`p` number of parameters

`n` number of rows

`q_post` A list of the fitted posterior Q family at iteration t.

`prior` Prior for the GMM parameters.

Value

elbo the Evidence Lower Bound at iteration t following E and M steps.

`VB_GMM_Expectation` *Variational Bayes Expectation step*

Description

Variational Bayes Expectation step

Usage

```
VB_GMM_Expectation(X, n, q_post)
```

Arguments

`X` n x p data matrix (or data frame that will be converted to a matrix).

`n` number of rows in X.

`q_post` A list of the fitted posterior Q family at iteration t-1.

Value

A list of the fitted posterior Q family after the E step at iteration t.

VB_GMM_Init *Initialise the variational parameters and the hyper parameters*

Description

Initialise the variational parameters and the hyper parameters

Usage

```
VB_GMM_Init(X, k, n, prior, init, initParams)
```

Arguments

X	n x p data matrix (or data frame that will be converted to a matrix).
k	guess for the number of mixture components.
n	number of rows in X.
prior	Prior for the GMM parameters.
init	initialize the clusters c("random", "kmeans", "dbscan")
initParams	initialization parameters for dbscan. NULL if dbscan not selected for init.

Value

A list of the initially fitted posterior Q family

VB_GMM_Maximisation *Variational Bayes Maximisation step*

Description

Variational Bayes Maximisation step

Usage

```
VB_GMM_Maximisation(X, q_post, prior)
```

Arguments

X	n x p data matrix (or data frame that will be converted to a matrix).
q_post	A list of the fitted posterior Q family at iteration t-1.
prior	Prior for the GMM parameters.

Value

A list of the fitted posterior Q family after the M step at iteration t.

Index