

# Package ‘vbmp’

May 11, 2024

**Type** Package

**Title** Variational Bayesian Multinomial Probit Regression

**Version** 1.72.0

**Author** Nicola Lama <nicola.lama@unina2.it>, Mark Girolami  
<girolami@dcs.gla.ac.uk>

**Maintainer** Nicola Lama <nicola.lama@unina2.it>

**Description** Variational Bayesian Multinomial Probit Regression with Gaussian Process Priors. It estimates class membership posterior probability employing variational and sparse approximation to the full posterior. This software also incorporates feature weighting by means of Automatic Relevance Determination.

**License** GPL (>= 2)

**Suggests** Biobase (>= 2.5.5), statmod

**LazyLoad** Yes

**Depends** R (>= 2.10)

**URL** <http://bioinformatics.oxfordjournals.org/cgi/content/short/btm535v1>

**biocViews** Classification

**git\_url** <https://git.bioconductor.org/packages/vbmp>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** 974fc8a

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-10

## Contents

|                     |   |
|---------------------|---|
| BRCA12 . . . . .    | 2 |
| covParams . . . . . | 3 |

|                           |   |
|---------------------------|---|
| lowerBound . . . . .      | 3 |
| plotDiagnostics . . . . . | 4 |
| predClass . . . . .       | 4 |
| predError . . . . .       | 5 |
| predictCPP . . . . .      | 5 |
| predLik . . . . .         | 6 |
| vbmp . . . . .            | 6 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>11</b> |
|--------------|-----------|

---

|        |                            |
|--------|----------------------------|
| BRCA12 | <i>BRCA tumour dataset</i> |
|--------|----------------------------|

---

## Description

This data set gives the gene expression values of 30 breast cancer patients. Short-term primary fibroblast cultures were established from skin biopsies from 10 BRCA1 and 10 BRCA2 mutation carriers and 10 controls.

## Usage

BRCA12

## Format

ExpressionSet object containing 8080 genes x 30 pts. Case and controls are specified in Target.class of phenoData.

## Source

The Institute of Cancer Research, Sutton, Surrey, UK

## References

Zsofia Kote-Jarai et al. *Accurate Prediction of BRCA1 and BRCA2 Heterozygous Genotype Using Expression Profiling after Induced DNA Damage*, Clin Cancer Res 2006;12(13)

---

|           |   |
|-----------|---|
| covParams | <i>VBMP covariance functions parameters</i> |
|-----------|---|

---

**Description**

Returns the value of the covariance functions parameters (theta).

**Usage**

```
covParams(obj)
```

**Arguments**

obj                    an object inheriting from class VBMP .obj, usually the result of a call to vbmp

**See Also**

See Also as [vbmp](#)

---

|            |                                  |
|------------|----------------------------------|
| lowerBound | <i>VBMP Lower bound estimate</i> |
|------------|----------------------------------|

---

**Description**

Returns the lower bound estimates for the VBMP fitted model.

**Usage**

```
lowerBound(obj)
```

**Arguments**

obj                    an object inheriting from class VBMP .obj, usually the result of a call to vbmp

**See Also**

See Also as [vbmp](#)

plotDiagnostics

*VBMP covariance diagnostics plot*

---

**Description**

plot the evolution of convergence diagnostics: lower-bound, predictive likelihood, out-of-bound test error and theta params (when inferred)

**Usage**

```
plotDiagnostics(obj)
```

**Arguments**

obj                    an object inheriting from class VBMP.obj, usually the result of a call to vbmp

**See Also**

See Also as [vbmp](#)

---

predClass

*VBMP Predicted class values*

---

**Description**

Predicted class targets of test dataset.

**Usage**

```
predClass(obj)
```

**Arguments**

obj                    an object inheriting from class VBMP.obj, usually the result of a call to vbmp

**See Also**

See Also as [vbmp](#)

---

|           |  |
|-----------|--|
| predError | <i>Out-of-Sample VBMP Prediction error</i> |
|-----------|--|

---

**Description**

Out-of-Sample Percent Prediction error estimate (0-1 error loss).

**Usage**

```
predError(obj)
```

**Arguments**

obj                    an object inheriting from class VBMP .obj, usually the result of a call to vbmp

**See Also**

See Also as [vbmp](#)

---

|            |  |
|------------|--|
| predictCPP | <i>VBMP predict functions parameters</i> |
|------------|--|

---

**Description**

Obtains estimates of class posterior probabilities from a fitted VBMP object

**Usage**

```
predictCPP(obj, X.TEST=NULL)
```

**Arguments**

obj                    an object inheriting from class VBMP .obj, usually the result of a call to vbmp  
X.TEST                optionally, matrix in which to look for variables with which to predict. If omitted, the fitted predictors are used.

**See Also**

See Also as [vbmp](#)

---

|         |  |
|---------|--|
| predLik | <i>VBMP predictive likelihood estimate</i> |
|---------|--|

---

**Description**

Returns the predictive likelihood estimate for the VBMP fitted model.

**Usage**

```
predLik(obj)
```

**Arguments**

`obj` an object inheriting from class `VBMP`. `obj`, usually the result of a call to `vbmp`

**See Also**

See Also as [vbmp](#)

---

|      |   |
|------|---|
| vbmp | <i>Variational Bayesian Multinomial Probit Regression with Gaussian Process Priors.</i> |
|------|---|

---

**Description**

Used to fit a Multinomial Probit Regression model, specified by giving the matrix design `X`, the associated response variables `t.class`, kernel type and covariate scaling parameters. Covariance parameters can be inferred from the data.

**Usage**

```
vbmp(X, t.class, X.TEST, t.class.TEST, theta, control = list())
```

**Arguments**

|                           |   |
|---------------------------|---|
| <code>X</code>            | Feature matrix for parameter 'estimation'   |
| <code>t.class</code>      | Target values, integer number used for class labels.                              |
| <code>X.TEST</code>       | Feature matrix to compute out-of-sample (test) prediction errors and likelihoods  |
| <code>t.class.TEST</code> | Target values for test data   |
| <code>theta</code>        | The covariance function parameters (e.g. scaling coefficients for each dimension) |
| <code>control</code>      | A list of control parameters. See Details   |

## Details

In this implementation a single covariance function is shared across all classes. Compute the predictive posteriors on the test set and the associated likelihood and test errors at each iteration.

The control argument is a list that can supply any of the following components:

**InfoLevel** 0 to suppress tracing (> 0 to print different levels of monitoring information)

**sFILE.TRACE** File name where to redirect output (default NULL)

**bThetaEstimate** if covariance parameter estimation switched on. Defaults to FALSE (switched off)

**sKernelType** Kernel function used in training and predicting. Currently implemented kernels are Gaussian ("gauss"), Cauchy ("cauchy"), Laplace ("laplace"), Polynomial ("poly"), Homogeneous polynomial ("hpoly"), 'Thin-plate' spline ("tps"), 'linear' spline ("lsp") and Inner product("iproduct"). Defaults to "gauss".

**maxIts** Maximum number of variational EM steps to take. Defaults to 50.

**Thresh** Convergence threshold on marginal likelihood lowerbound. Defaults to 1e-4.

**method** Integral computation method: "quadrature" (Gaussian quadrature) or "classic"(simple sampler). Defaults to "quadrature".

**nNodesQuad** Number of nodes used for quadrature. Defaults to 49.

**nSampsTG** Number of samples used in obtaining mean of truncated Gaussian. Defaults to 1000.

**nSampsIS** Number of samples used in the importance sampler. Defaults to 1000.

**nSmallNo** Small number used to prevent numerical problems (ill-conditioned covariance matrix). Defaults to 1e-10.

**parGammaTau,parGammaSigma** The location and scale parameters of the Gamma prior over covaraince params. Default to 1e-6.

**bMonitor** TRUE to collect monitor convergence diagnostics at each iteration. Defaults to FALSE.

**bPlotFitting** TRUE to plot test performance results at each iteration during model estimation (if TRUE it forces bMonitor to TRUE). Defaults to FALSE.

## Value

vbmp returns an object of class "VBMP.obj". An object of class "VBMP.obj" is a list containing at least the following components:

|             |  |
|-------------|--|
| Kc          | Number of classes  |
| Ptest       | Matrix of multinomial class predictive posterior probabilities for the test data         |
| X           | Feature matrix   |
| invPHI      | Inverse of the Kernel matrix   |
| Y           | Matrix of auxiliary variables  |
| M           | Matrix of GP random variables  |
| theta       | covariance kernel hyperparameters (estimates computed during model fitting, if inferred) |
| sKernelType | Kernel function used in training and predicting  |

|             |  |
|-------------|--|
| Test.Err    | Out-of-Sample Percent Prediction error estimates computed during model fitting (0-1 error loss). |
| PL          | Predictive Likelihood estimates computed during model fitting                                    |
| LOWER.BOUND | Lower bound estimates computed during model fitting  |

**Author(s)**

N Lama <nicola.lama@unina2.it>, MA Girolami <girolami@dcs.gla.ac.uk>

**References**

Girolami M, Rogers S, *Variational Bayesian Multinomial Probit Regression with Gaussian Process Priors*, Neural Computation 18, 1790-1817 (2006). Lama N, Girolami M *vbmp: Variational Bayesian Multinomial Probit Regression for multi-class classification in R*, Bioinformatics 24(1):135-136 (2008). <http://bioinformatics.oxfordjournals.org/cgi/content/short/btm535v1>

**See Also**

See Also as [predictCPP](#), [covParams](#), [lowerBound](#), [predError](#), [predLik](#), [predClass](#)

**Examples**

```
## -----
## EXAMPLE 1 - Theta estimate with synthetic data
## -----
## Samples of 2-D data points drawn from three nonlinearly separable
## classes which take the form of two annular rings and one zero-centered
## Gaussian are used in this little illustrative example.
genSample <- function(n, noiseVar=0) {
  ## class 1 and 2 (x ~ U(0,1))
  u <- 4. * matrix(runif(2*n), nrow=n, ncol=2) - 2.;
  i <- which(((u[, 1]^2 + u[, 2]^2) > .1) & ((u[, 1]^2 + u[, 2]^2) < .5) );
  j <- which(((u[, 1]^2 + u[, 2]^2) > .6) & ((u[, 1]^2 + u[, 2]^2) < 1) );
  X <- u[c(i, j),];
  t.class <- c(rep(1, length(i)),rep(2, length(j)));
  ## class 3 (x ~ N(0,1))
  x <- 0.1 * matrix(rnorm(2*length(i)), ncol=2, nrow=length(i) );
  k <- which((x[, 1]^2 + x[, 2]^2) < 0.1);
  X <- rbind(X, x[k, ]);
  t.class <- c(t.class, rep(3, length(k)));
  ## add random coloumns
  if (noiseVar>0) X <- cbind(X, matrix(rnorm(noiseVar*nrow(X)), ncol=noiseVar, nrow=nrow(X)));
  structure( list( t.class=t.class, X=X), class="MultiNoisyData");
}

set.seed(123); ## Init random number generator

## Generate training and test samples as an independent
## test set to assess out-of-sample prediction error
## and predictive likelihoods.
nNoisyInputs <- 0;      ## number of additional noisy input parameters
```



```

Ntest <- Ntrain <- 500; ## sample sizes
dataXt.train <- genSample(Ntrain, nNoisyInputs);
dataXt.test <- genSample(Ntest, nNoisyInputs);

## Not run:
theta <- runif(ncol(dataXt.train$X));
res <- vbmp( dataXt.train$X, dataXt.train$t.class,
            dataXt.test$X, dataXt.test$t.class, theta,
            control=list(bThetaEstimate = T,
                        bPlotFitting=T, maxIts=50));

## End(Not run)

## set theta params (previously estimated)
theta <- c(0.09488309, 0.16141604);
## Fit the vbmp
res <- vbmp( dataXt.train$X, dataXt.train$t.class,
            dataXt.test$X, dataXt.test$t.class, theta,
            control=list(maxIts=5));
## print out-of-sample error estimate
predError(res);

## Not run:
## -----
## EXAMPLE 2 - BRCA12 genomic data
## -----
## Leave-one-out (LOO) cross-validation prediction error of the probabilistic
## Gaussian process classifier used in Zsofia Kote-Jarai et al.
## Clin Cancer Res 2006;12(13);3896-3901

if(any(installed.packages()[,1]=="Biobase")) {
  library("Biobase");
  data("BRCA12");
  brca.y <- BRCA12$Target.class;
  brca.x <- t(exprs(BRCA12));
} else {
  print("Deprecated....");
  load(url("http://www.dcs.gla.ac.uk/people/personal/girolami/pubs_2005/VBGP/BRCA12.RData"));
  brca.y <- as.numeric(BRCA12$y);
  brca.x <- as.matrix(BRCA12[, -1]);
}

sKernelType <- "iproduct"; ## Covariance function type
Thresh <- 1e-8; ## Iteration threshold
InfoLevel <- 1;
theta <- rep(1.0, ncol(brca.x));
ITER.THETA <- 24;
n <- nrow(brca.x) ;
Kfold <- n; # number of folds , if equal to n then LOO
samps <- sample(rep(1:Kfold, length=n), n, replace=FALSE);
res <- rep(NA, n);
print(paste("LOO crossvalidation started..... (" ,n,"steps)"));
for (x in 1:Kfold) {

```

```
cat(paste(x, " ", sep="")); flush.console();
resX <- vbmp( brca.x[samps!=x,], brca.y[samps!=x],
             brca.x[samps==x,], brca.y[samps==x],
             theta, control=list(bThetaEstimate=F,
                                 bPlotFitting=F, maxIts=ITER.THETA,
                                 sKernelType=sKernelType, Thresh=Thresh));
res[samps==x] <- predClass(resX);
}
print("(end)");
print(paste("Crossvalidated error rate", round(sum(res!=brca.y)/n,2)));

## End(Not run)
```

# Index

\* **datasets**

BRCA12, 2

\* **models**

vbmp, 6

\* **utilities**

covParams, 3

lowerBound, 3

plotDiagnostics, 4

predClass, 4

predError, 5

predictCPP, 5

predLik, 6

BRCA12, 2

covParams, 3, 8

lowerBound, 3, 8

plotDiagnostics, 4

predClass, 4, 8

predError, 5, 8

predictCPP, 5, 8

predLik, 6, 8

vbmp, 3–6, 6