Package ‘CVEK’

July 27, 2018

Title Cross-Validated Kernel Ensemble

Version 0.2-0

Date 2018-06-27

Description Using a library of base kernels, it learns a proper generating function from data by directly minimizing the ensemble model’s error, and tests whether the data is generated by the RKHS under the null hypothesis.

Depends R (>= 3.0.1), mvtnorm, MASS, psych, limSolve

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Wenying Deng [aut, cre],
Jeremiah Zhe Liu [ctb]

Maintainer Wenying Deng <wdeng@hsph.harvard.edu>

R topics documented:

---

compute_info *Computing Information Matrices*

Description

Compute information matrices based on block matrices.

Usage

```
compute_info(P0_mat, mat_del = NULL, mat_sigma2 = NULL, mat_tau = NULL)
```

Arguments

P0_mat (matrix, n*n) Scale projection matrix under REML.

mat_del (matrix, n*n) Derivative of the scale covariance matrix of Y with respect to delta.

mat_sigma2 (matrix, n*n) Derivative of the scale covariance matrix of Y with respect to sigma2.

mat_tau (matrix, n*n) Derivative of the scale covariance matrix of Y with respect to tau.

Details

This function gives the information value of the interaction strength.

Value

I0 (matrix, n*n) The computed information value.

Author(s)

Wenying Deng

References

Arnab Maity and Xihong Lin. Powerful tests for detecting a gene effect in the presence of possible gene-gene interactions using garrote kernel machines. December 2011.

Examples

```
I0 <- compute_info(P0_mat, mat_del = drV0_del,
mat_sigma2 = drV0_sigma2, mat_tau = drV0_tau)
```

compute_stat *Computing Score Test Statistics.*

Description

Compute score test statistics.

Usage

```
compute_stat(n, Y, X12, beta0, sigma2_hat, tau_hat, K_gpr)
```

Arguments

n	(integer) A numeric number specifying the number of observations.
Y	(vector of length n) Reponses of the dataframe.
X12	(dataframe, n*(p1*p2)) The interaction items of first and second types of factors in the dataframe.
beta0	(numeric) Estimated bias of the model.
sigma2_hat	(numeric) The estimated noise of the fixed effects.
tau_hat	(numeric) The estimated noise of the random effects.
K_gpr	(matrix, n*n) Estimated ensemble kernel matrix.

Details

The test statistic is distributed as a scaled Chi-squared distribution.

Value

test_stat	(numeric) The computed test statistic.

Author(s)

Wenying Deng

References

Arnab Maity and Xihong Lin. Powerful tests for detecting a gene effect in the presence of possible gene-gene interactions using garrote kernel machines. December 2011.

Examples

```
compute_stat(n = 100, Y, X12, beta0, sigma2_hat, tau_hat, K_gpr)
```

define_model *Defining the Model*

Description

Give the complete formula and generate the expected kernel library.

Usage

```
define_model(formula, label_names, data, kern_par)
```

Arguments

`formula` (formula) A symbolic description of the model to be fitted.

`label_names` (list) A character string indicating all the interior variables included in each predictor. See Details.

`data` (dataframe, n*P) A dataframe to be fitted. See Details.

`kern_par` (dataframe, K*4) A dataframe indicating the parameters of base kernels to be created. See Details.

Details

It processes data based on formula and label_names and creates a kernel library according to the parameters given in kern_par.

* label_names: for two subgroups with sizes p1 and p2 respectively, label_names contains two elements. The length of the first element is p1, indicating the names of p1 interiors variables, and the length of second one is p2, indicating the names of p2 interiors variables.

* data: for a data with n observations and P=p1+p2 variables (with sub-groups of sizes (p1, p2)), the dimension of dataframe is n*P. All entries should be numeric and the column name of response is "Y", while the column names of P variables are the ones from label_names.

* kern_par: for a library of K kernels, the dimension of this dataframe is K*4. Each row represents a kernel. The first column is method, with entries of character class. The second is Sigma, with entries of matrix class, indicating the covariance matrix for neural network kernel (default=0). The third and the fourth are l and p respectively, both with entries of numeric class.

Value

`Y` (vector of length n) Reponses of the dataframe.

`X1` (dataframe, n*p1) The first type of factor in the dataframe (could contains several subfactors).

`X2` (dataframe, n*p2) The second type of factor in the dataframe (could contains several subfactors).

`kern_list` (list of length K) A list of kernel functions given by user.

Author(s)

Wenying Deng

See Also

method: `generate_kernel`

Examples

```
kern_par <- data.frame(method = c("rbf", "polynomial", "matern"),
Sigma = rep(0, 3), l = c(.5, 1, 1.5), p = 1:3)
kern_par$method <- as.character(kern_par$method)
define_model(formula = Y ~ X1 + X2,
label_names = list(X1 = c("x1", "x2"), X2 = c("x3", "x4")),
data = dora, kern_par)
```

`ensemble` *Estimating Ensemble Kernel Matrices*

Description

Give a list of estimated kernel matrices and their weights.

Usage

```
ensemble(n, kern_size, strategy, beta, error_mat, A_hat)
```

Arguments

`n`	(integer) A numeric number specifying the number of observations.
`kern_size`	(integer, =K) A numeric number specifying the number of kernels in the kernel library.
`strategy`	(character) A character string indicating which ensemble strategy is to be used.
`beta`	(numeric) A numeric value specifying the parameter when strategy = "exp".
`error_mat`	(matrix, n*K) A n*kern_size matrix indicating errors.
`A_hat`	(list of length K) A list of projection matrices for every kernels in the kernel library.

Details

There are three ensemble strategies available here:

Empirical Risk Minimization

After obtaining the estimated errors $\{\hat{\epsilon}_d\}_{d=1}^D$, we estimate the ensemble weights $u = \{u_d\}_{d=1}^D$ such that it minimizes the overall error

$$\hat{u} = u \in \Delta argmin \parallel \sum_{d=1}^D u_d \hat{\epsilon}_d \parallel^2 \quad where\ \Delta = \{u|u \geq 0, \parallel u \parallel_1 = 1\}$$

Then produce the final ensemble prediction:

$$\hat{h} = \sum_{d=1}^D \hat{u}_d h_d = \sum_{d=1}^D \hat{u}_d A_{d,\hat{\lambda}_d} y = \hat{A} y$$

where $\hat{A} = \sum_{d=1}^D \hat{u}_d A_{d,\hat{\lambda}_d}$ is the ensemble matrix.

Simple Averaging

Motivated by existing literature in omnibus kernel, we propose another way to obtain the ensemble matrix by simply choosing unsupervised weights $u_d = 1/D$ for $d = 1, 2, ...D$.

Exponential Weighting

Additionally, another scholar gives a new strategy to calculate weights based on the estimated errors $\{\hat{\epsilon}_d\}_{d=1}^D$.

$$u_d(\beta) = \frac{exp(- \parallel \hat{\epsilon}_d \parallel_2^2 / \beta)}{\sum_{d=1}^D exp(- \parallel \hat{\epsilon}_d \parallel_2^2 / \beta)}$$

Value

`A_est` (matrix, n*n) A list of estimated kernel matrices.

`u_hat` (vector of length K) A vector of weights of the kernels in the library.

Author(s)

Wenying Deng

References

Jeremiah Zhe Liu and Brent Coull. Robust Hypothesis Test for Nonlinear Effect with Gaus- sian Processes. October 2017.

Xiang Zhan, Anna Plantinga, Ni Zhao, and Michael C. Wu. A fast small-sample kernel independence test for microbiome community-level association analysis. December 2017.

Arnak S. Dalalyan and Alexandre B. Tsybakov. Aggregation by Exponential Weighting and Sharp Oracle Inequalities. In Learning Theory, Lecture Notes in Computer Science, pages 97– 111. Springer, Berlin, Heidelberg, June 2007.

See Also

mode: `tuning`

Examples

```
ensemble(n = 100, kern_size = 3, strategy = "erm", beta = 1, error_mat, A_hat)
```

`estimate_base` *Estimating Projection Matrices*

Description

Calculate the estiamted projection matrices for every kernels in the kernel library.

Usage

```
estimate_base(n, kern_size, Y, X1, X2, kern_list, mode, lambda)
```

Arguments

`n` (integer) A numeric number specifying the number of observations.

`kern_size` (integer, =K) A numeric number specifying the number of kernels in the kernel library.

`Y` (vector of length n) Reponses of the dataframe.

`X1` (dataframe, n*p1) The first type of factor in the dataframe (could contains several subfactors).

`X2`	(dataframe, n*p2) The second type of factor in the dataframe (could contains several subfactors).
`kern_list`	(list of length K) A list of kernel functions given by user.
`mode`	(character) A character string indicating which tuning parameter criteria is to be used.
`lambda`	(numeric) A numeric string specifying the range of noise to be chosen. The lower limit of lambda must be above 0.

Details

For a given mode, this function return a list of projection matrices for every kernels in the kernel library and a n*kern_size matrix indicating errors.

Value

`A_hat`	(list of length K) A list of projection matrices for every kernels in the kernel library.
`error_mat`	(matrix, n*K) A n*kern_size matrix indicating errors.

Author(s)

Wenying Deng

References

Jeremiah Zhe Liu and Brent Coull. Robust Hypothesis Test for Nonlinear Effect with Gaus- sian Processes. October 2017.

Examples

```
estimate_base(n = 100, kern_size = 3, Y, X1, X2, kern_list,
mode = "loocv", lambda = exp(seq(-5, 5)))
```

`estimate_noise` *Estimating Noise*

Description

An implementation of Gaussian processes for estimating noise.

Usage

```
estimate_noise(Y, lambda_hat, beta_hat, alpha_hat, K_hat)
```

Arguments

`Y`	(vector of length n) Reponses of the dataframe.
`lambda_hat`	(numeric) The selected tuning parameter based on the estimated ensemble kernel matrix.
`beta_hat`	(numeric) Estimated bias of the model.
`alpha_hat`	(vector of length n) Estimated coefficients of the estimated ensemble kernel matrix.
`K_hat`	(matrix, n*n) Estimated ensemble kernel matrix.

Value

`sigma2_hat`	(numeric) The estimated noise of the fixed effects.

Author(s)

Wenying Deng

References

Jeremiah Zhe Liu and Brent Coull. Robust Hypothesis Test for Nonlinear Effect with Gaus- sian Processes. October 2017.

Examples

```
sigma2_hat <- estimate_noise(Y, lam, beta0, alpha0, K_gpr)
```

estimation *Conducting Gaussian Process Regression*

Description

Conduct gaussian process regression based on the estimated ensemble kernel matrix.

Usage

```
estimation(Y, X1, X2, kern_list, mode = "loocv", strategy = "erm",
  beta = 1, lambda = exp(seq(-5, 5)))
```

Arguments

`Y`	(vector of length n) Reponses of the dataframe.
`X1`	(dataframe, n*p1) The first type of factor in the dataframe (could contains several subfactors).
`X2`	(dataframe, n*p2) The second type of factor in the dataframe (could contains several subfactors).
`kern_list`	(list of length K) A list of kernel functions given by user.

`mode` (character) A character string indicating which tuning parameter criteria is to be used.

`strategy` (character) A character string indicating which ensemble strategy is to be used.

`beta` (numeric) A numeric value specifying the parameter when strategy = "exp".

`lambda` (numeric) A numeric string specifying the range of noise to be chosen. The lower limit of lambda must be above 0.

Details

After obtaining the ensemble kernel matrix, we can calculate the outpur of gaussian process regression, the solution is given by

$$\hat{\beta} = [1^T(K+\lambda I)^{-1}1]^{-1}1^T(K+\lambda I)^{-1}y$$

$$\hat{\alpha} = (K+\lambda I)^{-1}(y-\hat{\beta}1)$$

where $\beta = intercept$.

Value

`lam` (numeric) The selected tuning parameter based on the estimated ensemble kernel matrix.

`intercept` (numeric) Estimated bias of the model.

`alpha` (vector of length n) Estimated coefficients of the estimated ensemble kernel matrix.

`K` (matrix, n*n) Estimated ensemble kernel matrix.

`u_hat` (vector of length K) A vector of weights of the kernels in the library.

Author(s)

Wenying Deng

See Also

strategy: `ensemble`

Examples

```
estimation(Y, X1, X2, kern_list, mode = ”loocv”, strategy = ”erm”,
beta = 1, lambda = exp(seq(-5, 5)))
```

generate_data *Generating Original Data*

Description

Generate original data based on specific kernels.

Usage

```
generate_data(n, label_names, method = "rbf", int_effect = 0, l = 1,
  p = 2, eps = 0.01)
```

Arguments

`n` (integer) A numeric number specifying the number of observations.

`label_names` (list) A character string indicating all the interior variables included in each predictor.

`method` (character) A character string indicating which kernel is to be computed.

`int_effect` (numeric) A numeric number specifying the size of interaction.

`l` (numeric) A numeric number indicating the hyperparameter (flexibility) of a specific kernel.

`p` (integer) For polynomial, p is the power; for matern, v = p + 1 / 2; for rational, alpha = p.

`eps` (numeric) A numeric number indicating the size of noise.

Details

This function generates with a specific kernel. The argument int_effect represents the strength of interaction relative to the main effect since all sampled functions have been standardized to have unit norm.

Value

`data, n*P` (dataframe) A dataframe to be fitted.

Author(s)

Wenying Deng

Examples

```
data <- generate_data(n = 100, label_names =
list(X1 = c("x1", "x2"), X2 = c("x3", "x4")),
method = "rbf", int_effect = 0, l = 1, p = 2, eps = .01)
```

generate_formula *From Vectors to Single Variables*

Description

Transform format of predictors from vectors to single variables.

Usage

```
generate_formula(formula, label_names)
```

Arguments

`formula`	(formula) A symbolic description of the model to be fitted.
`label_names`	(list) A character string indicating all the interior variables included in each predictor.

Value

`generic_formula`	(formula) A symbolic description of the model written in single variables format.
`length_main`	(integer) A numeric value indicating the length of main effects.

Author(s)

Wenying Deng

Examples

```
generic_formula0 <- generate_formula(formula = Y ~ X1 + X2,
label_names = list(X1 = c("x1", "x2"), X2 = c("x3", "x4")))
```

generate_kernel *Generating A Single Kernel*

Description

Generate kernels for the kernel library.

Usage

```
generate_kernel(method = "rbf", Sigma = 0, l = 1, p = 2)
```

Arguments

`method` (character) A character string indicating which kernel is to be computed.

`Sigma` (matrix) The covariance matrix for neural network kernel.

`l` (numeric) A numeric number indicating the hyperparameter (flexibility) of a specific kernel.

`p` (integer) For polynomial, p is the power; for matern, v = p + 1 / 2; for rational, alpha = p.

Details

There are seven kinds of kernel available here. For convenience, we define $r =\mid x - x' \mid$.

Gaussian RBF Kernels

$$k_{SE}(r) = exp\Big(-\frac{r^2}{2l^2}\Big)$$

Matern Kernels

$$k_{Matern}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\Big(\frac{\sqrt{2\nu r}}{l}\Big)^{\nu} K_{\nu}\Big(\frac{\sqrt{2\nu r}}{l}\Big)$$

Rational Quadratic Kernels

$$k_{RQ}(r) = \Big(1 + \frac{r^2}{2\alpha l^2}\Big)^{-\alpha}$$

Polynomial Kernels

$$k(x, x') = (x \cdot x')^p$$

We have intercept kernel when $p = 0$, and linear kernel when $p = 1$.

Neural Network Kernels

$$k_{NN}(x, x') = \frac{2}{\pi} sin^{-1}\Big(\frac{2\tilde{x}^T \Sigma \tilde{x}'}{\sqrt{(1 + 2\tilde{x}^T \Sigma \tilde{x})(1 + 2\tilde{x}'^T \Sigma \tilde{x}')}}\Big)$$

Value

`kern` (function) A function indicating the generated kernel.

Author(s)

Wenying Deng

References

The MIT Press. Gaussian Processes for Machine Learning, 2006.

Examples

```
kern_list <- list()
for (d in 1:nrow(kern_par)) {
  kern_list[[d]] <- generate_kernel(kern_par[d, ]$method,
                                    kern_par[d, ]$Sigma,
                                    kern_par[d, ]$l,
                                    kern_par[d, ]$p)
}
```

`testing` *Conducting Score Tests for Interaction*

Description

Conduct score tests comparing a fitted model and a more general alternative model.

Usage

```
testing(formula_int, label_names, Y, X1, X2, kern_list, mode = "loocv",
  strategy = "erm", beta = 1, test = "boot", lambda = exp(seq(-5, 5)),
  B = 100)
```

Arguments

`formula_int`	(formula) A symbolic description of the model with interaction.
`label_names`	(list) A character string indicating all the interior variables included in each predictor.
`Y`	(vector of length n) Reponses of the dataframe.
`X1`	(dataframe, n*p1) The first type of factor in the dataframe (could contains several subfactors).
`X2`	(dataframe, n*p2) The second type of factor in the dataframe (could contains several subfactors).
`kern_list`	(list of length K) A list of kernel functions given by user.
`mode`	(character) A character string indicating which tuning parameter criteria is to be used.
`strategy`	(character) A character string indicating which ensemble strategy is to be used.
`beta`	(numeric) A numeric value specifying the parameter when strategy = "exp".
`test`	(character) A character string indicating which test is to be used.
`lambda`	(numeric) A numeric string specifying the range of noise to be chosen. The lower limit of lambda must be above 0.
`B`	(integer) A numeric value indicating times of resampling w hen test = "boot".

Details

There are two tests available here:

Asymptotic Test

This is based on the classical variance component test to construct a testing procedure for the hypothesis about Gaussian process function.

Bootstrap Test

When it comes to small sample size, we can use bootstrap test instead, which can give valid tests with moderate sample sizes and requires similar computational effort to a permutation test.

Value

`pvalue`	(numeric) p-value of the test.

Author(s)

Wenying Deng

References

Xihong Lin. Variance component testing in generalised linear models with random effects. June 1997.

Arnab Maity and Xihong Lin. Powerful tests for detecting a gene effect in the presence of possible gene-gene interactions using garrote kernel machines. December 2011.

Petra Bu zˇkova , Thomas Lumley, and Kenneth Rice. Permutation and parametric bootstrap tests for gene-gene and gene-environment interactions. January 2011.

See Also

method: `generate_kernel`

mode: `tuning`

strategy: `ensemble`

Examples

```
testing(formula_int = Y ~ X1 * X2,
label_names = list(X1 = c("x1", "x2"), X2 = c("x3", "x4")),
Y, X1, X2, kern_list, mode = "loocv", strategy = "erm",
beta = 1, test = "boot", lambda = exp(seq(-5, 5)), B = 100)
```

`tuning` *Calculating Tuning Parameters*

Description

Calculate tuning parameters based on given criteria.

Usage

```
tuning(Y, K_mat, mode, lambda)
```

Arguments

`Y`	(vector of length n) Reponses of the dataframe.
`K_mat`	(matrix, n*n) Estimated ensemble kernel matrix.
`mode`	(character) A character string indicating which tuning parameter criteria is to be used.
`lambda`	(numeric) A numeric string specifying the range of noise to be chosen. The lower limit of lambda must be above 0.

Details

There are four tuning parameter selections here:

leave-one-out Cross Validation

$$\lambda_{n-CV} = \lambda \in \Lambda argmin \left\{ log\ y^{\star T}[I - diag(A_\lambda) - \frac{1}{n}I]^{-1}(I - A_\lambda)^2[I - diag(A_\lambda) - \frac{1}{n}I]^{-1}y^\star \right\}$$

Akaike Information Criteria

$$\lambda_{AICc} = \lambda \in \Lambda argmin \Big\{ log\ y^{\star T}(I - A_\lambda)^2 y^\star + \frac{2[tr(A_\lambda) + 2]}{n - tr(A_\lambda) - 3} \Big\}$$

Generalized Cross Validation

$$\lambda_{GCVc} = \lambda \in \Lambda argmin \Big\{ log\ y^{\star T}(I - A_\lambda)^2 y^\star - 2log[1 - \frac{tr(A_\lambda)}{n} - \frac{2}{n}]_+ \Big\}$$

Generalized Maximum Profile Marginal Likelihood

$$\lambda_{GMPML} = \lambda \in \Lambda argmin \Big\{ log\ y^{\star T}(I - A_\lambda)y^\star - \frac{1}{n-1} log \mid I - A_\lambda \mid \Big\}$$

Value

`lambda0`	(numeric) The estimated tuning parameter.

Author(s)

Wenying Deng

References

Philip S. Boonstra, Bhramar Mukherjee, and Jeremy M. G. Taylor. A Small-Sample Choice of the Tuning Parameter in Ridge Regression. July 2015.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer Series in Statistics. Springer- Verlag, New York, 2 edition, 2009.

Hirotogu Akaike. Information Theory and an Extension of the Maximum Likelihood Princi- ple. In Selected Papers of Hirotugu Akaike, Springer Series in Statistics, pages 199–213. Springer, New York, NY, 1998.

Clifford M. Hurvich and Chih-Ling Tsai. Regression and time series model selection in small samples. June 1989.

Hurvich Clifford M., Simonoff Jeffrey S., and Tsai Chih-Ling. Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. January 2002.

Examples

```
lambda0 <- tuning(Y, K_mat = K_hat,
mode = "loocv", lambda = exp(seq(-5, 5)))
```