Package ‘CVEK’

December 18, 2020

Title Cross-Validated Kernel Ensemble

Version 0.1-2

Date 2020-11-27

Description Implementation of Cross-Validated Kernel Ensemble (CVEK), a flexible modeling framework for robust nonlinear regression and hypothesis testing based on ensemble learning with kernel-ridge estimators (Jeremiah et al. (2017) <arXiv:1710.01406> and Wenying et al. (2018) <arXiv:1811.11025>). It allows user to conduct nonlinear regression with minimal assumption on the function form by aggregating nonlinear models generated from a diverse collection of kernel families. It also provides utilities to test for the estimated nonlinear effect under this ensemble estimator, using either the asymptotic or the bootstrap version of a generalized score test.

Depends R (>= 3.6.0), MASS, limSolve

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests testthat, knitr, rmarkdown, ggplot2, ggrepel

VignetteBuilder knitr

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2020-12-18 16:10:18 UTC

R topics documented:

`cvek` *Conducting Cross-validated Kernel Ensemble*

Description

Conducting Cross-validated Kernel Ensemble based on user-specified formula.

Usage

```
cvek(
  formula,
  kern_func_list,
  data,
  formula_test = NULL,
  mode = "loocv",
  strategy = "stack",
  beta_exp = 1,
  lambda = exp(seq(-10, 5)),
  test = "boot",
```

```
  alt_kernel_type = "linear",
  B = 100,
  verbose = FALSE
)
```

Arguments

formula	(formula) A user-supplied formula for the null model. Should contain at least one kernel term.
kern_func_list	(list) A list of kernel functions in the kernel library.
data	(data.frame, n*d) A data.frame, list or environment (or object coercible by as.data.frame to a data.frame), containing the variables in formula. Neither a matrix nor an array will be accepted.
formula_test	(formula) A user-supplied formula indicating the alternative effect to test. All terms in the alternative mode must be specified as kernel terms.
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_exp	(numeric/character) A numeric value specifying the parameter when strategy = "exp" ensemble_exp.
lambda	(numeric) A numeric string specifying the range of tuning parameter to be chosen. The lower limit of lambda must be above 0.
test	(character) Type of hypothesis test to conduct. Must be either 'asymp' or 'boot'.
alt_kernel_type	(character) Type of alternative kernel effect to consider. Must be either "linear" or "ensemble".
B	(numeric) Number of bootstrap samples.
verbose	(logical) Whether to print additional messages.

Details

Perform Cross-validated Kernel Ensemble and optionally test for kernel effect based on user-specified formula.

Value

lambda	(numeric) The selected tuning parameter based on the estimated ensemble kernel matrix.
beta	(matrix, d_fixed*1) Fixed effect estimates.
alpha	(matrix, n*1) Kernel effect estimates.
K	(matrix, n*n) Estimated ensemble kernel matrix.
u_hat	(vector of length K) A vector of weights of the kernels in the library.
base_est	(list) The detailed estimation results of K kernels.
pvalue	(numeric) If formula_test is given, p-value of the test is returned.

Author(s)

Jeremiah Zhe Liu

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 z kova, Thomas Lumley, and Kenneth Rice. Permutation and parametric bootstrap tests for gene-gene and gene-environment interactions. January 2011.

See Also

estimation

method: generate_kernel

mode: tuning

strategy: ensemble

Examples

```
kern_par <- data.frame(method = rep("rbf", 3),
l = rep(3, 3), p = rep(2, 3),
stringsAsFactors = FALSE)
# define kernel library
kern_func_list <- define_library(kern_par)

n <- 10
d <- 4
formula <- y ~ x1 + x2 + k(x3, x4)
formula_test <- y ~ k(x1, x2) * k(x3, x4)
set.seed(1118)
data <- as.data.frame(matrix(
  rnorm(n * d),
  ncol = d,
  dimnames = list(NULL, paste0("x", 1:d))
))
beta_true <- c(1, .41, 2.37)
lnr_kern_func <- generate_kernel(method = "rbf", l = 3)
kern_effect_lnr <-
  parse_kernel_variable("k(x3, x4)", lnr_kern_func, data)
alpha_lnr_true <- rnorm(n)

data$y <- as.matrix(cbind(1, data[, c("x1", "x2")])) %*% beta_true +
  kern_effect_lnr %*% alpha_lnr_true

data_train <- data
```

```
pvalue <- cvek(formula,
               kern_func_list,
               data_train,
               formula_test,
               mode = "loocv",
               strategy = "stack",
               beta_exp = 1,
               lambda = exp(seq(-2, 2)),
               test = "asymp",
               alt_kernel_type = "linear",
               verbose = FALSE)$pvalue
```

`define_library` *Defining Kernel Library*

Description

Generate the expected kernel library based on user-specified dataframe.

Usage

```
define_library(kern_par = NULL)
```

Arguments

`kern_par`	(dataframe, K*3) A dataframe indicating the parameters of base kernels to fit kernel effect. See Details.

Details

It creates a kernel library according to the parameters given in kern_par.

* kern_par: for a library of K kernels, the dimension of this dataframe is K*3. Each row represents a kernel. The first column is method, with entries of character class. The second and the third are l and p respectively, both with entries of numeric class.

Value

`kern_func_list`	(list of length K) A list of kernel functions given by user. Will be overwritten to linear kernel if kern_par is NULL.

Author(s)

Wenying Deng

See Also

method: `generate_kernel`

ensemble *Estimating Ensemble Kernel Matrices*

Description

Give the ensemble projection matrix and weights of the kernels in the library.

Usage

```
ensemble(strategy, beta_exp, error_mat, A_hat)
```

Arguments

`strategy`	(character) A character string indicating which ensemble strategy is to be used.
`beta_exp`	(numeric/character) A numeric value specifying the parameter when strategy = "exp" `ensemble_exp`.
`error_mat`	(matrix, n*K) A n*K matrix indicating errors.
`A_hat`	(list of length K) A list of projection matrices to kernel space for each kernel in the kernel library.

Details

There are three ensemble strategies available here:

Empirical Risk Minimization (Stacking)

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

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

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) The ensemble projection matrix.
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 Gaussian 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

---

ensemble_avg — *Estimating Ensemble Kernel Matrices Using AVG*

Description

Give the ensemble projection matrix and weights of the kernels in the library using simple averaging.

Usage

```
ensemble_avg(beta_exp, error_mat, A_hat)
```

Arguments

beta_exp	(numeric/character) A numeric value specifying the parameter when strategy = "exp" ensemble_exp.
error_mat	(matrix, n*K) A n*K matrix indicating errors.
A_hat	(list of length K) A list of projection matrices to kernel space for each kernel in the kernel library.

Details

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 $d = 1, 2, ...D$.

Value

A_est	(matrix, n*n) The ensemble projection matrix.
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 Gaussian 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

`ensemble_exp` *Estimating Ensemble Kernel Matrices Using EXP*

Description

Give the ensemble projection matrix and weights of the kernels in the library using exponential weighting.

Usage

```
ensemble_exp(beta_exp, error_mat, A_hat)
```

Arguments

beta_exp	(numeric/character) A numeric value specifying the parameter when strategy = "exp". See Details.
error_mat	(matrix, n*K) A n*K matrix indicating errors.
A_hat	(list of length K) A list of projection matrices to kernel space for each kernel in the kernel library.

Details

Exponential Weighting

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

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

beta_exp

The value of beta_exp can be "min"=$min\{RSS\}_{d=1}^D/10$, "med"=$median\{RSS\}_{d=1}^D$, "max"=$max\{RSS\}_{d=1}^D*2$ and any other positive numeric number, where $\{RSS\}_{d=1}^D$ are the set of residual sum of squares of D base kernels.

Value

A_est	(matrix, n*n) The ensemble projection matrix.
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 Gaussian 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

---

ensemble_stack *Estimating Ensemble Kernel Matrices Using Stack*

Description

Give the ensemble projection matrix and weights of the kernels in the library using stacking.

Usage

```
ensemble_stack(beta_exp, error_mat, A_hat)
```

Arguments

- `beta_exp` (numeric/character) A numeric value specifying the parameter when strategy = "exp" `ensemble_exp`.
- `error_mat` (matrix, n*K) A n*K matrix indicating errors.
- `A_hat` (list of length K) A list of projection matrices to kernel space for each kernel in the kernel library.

Details

Empirical Risk Minimization (Stacking)

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

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

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.

Value

- `A_est` (matrix, n*n) The ensemble projection matrix.
- `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 Gaussian 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`

`estimate_ridge` *Estimating a Single Model*

Description

Estimating projection matrices and parameter estimates for a single model.

Usage

```
estimate_ridge(
  Y,
  X,
  K,
  lambda,
  compute_kernel_terms = TRUE,
  converge_thres = 1e-04
)
```

Arguments

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`K`	(list of matrices) A nested list of kernel term matrices, corresponding to each kernel term specified in the formula for a base kernel function in kern_func_list.
`lambda`	(numeric) A numeric string specifying the range of tuning parameter to be chosen. The lower limit of lambda must be above 0.
`compute_kernel_terms`	(logic) Whether to computing effect for each individual terms. If FALSE then only compute the overall effect.
`converge_thres`	(numeric) The convergence threshold for computing kernel terms.

Details

For a single model, we can calculate the output of gaussian process regression, the solution is given by

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

.

Value

`beta`	(matrix, d_fixed*1) Fixed effect estimates.
`alpha`	(matrix, n*k_terms) Kernel effect estimates for each kernel term.
`kern_term_mat`	(matrix, n*k_terms) Kernel effect for each kernel term.
`A_list`	(list of length k_terms) Projection matrices for each kernel term.
`proj_matrix`	(list of length 4) Estimated projection matrices, combined across kernel terms.

Author(s)

Wenying Deng

References

Andreas Buja, Trevor Hastie, and Robert Tibshirani. (1989) Linear Smoothers and Additive Models. Ann. Statist. Volume 17, Number 2, 453-510.

`estimation` *Conducting Gaussian Process Regression*

Description

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

Usage

```
estimation(
  Y,
  X,
  K_list = NULL,
  mode = "loocv",
  strategy = "stack",
  beta_exp = 1,
  lambda = exp(seq(-10, 5)),
  ...
)
```

Arguments

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`K_list`	(list of matrices) A nested list of kernel term matrices. The first level corresponds to each base kernel function in kern_func_list, the second level corresponds to each kernel term specified in the formula.
`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_exp`	(numeric/character) A numeric value specifying the parameter when strategy = "exp" `ensemble_exp`.
`lambda`	(numeric) A numeric string specifying the range of tuning parameter to be chosen. The lower limit of lambda must be above 0.
`...`	Additional parameters to pass to estimate_ridge.

Details

After obtaining the ensemble kernel matrix, we can calculate the output of Gaussian process regression.

Value

`lambda`	(numeric) The selected tuning parameter based on the estimated ensemble kernel matrix.
`beta`	(matrix, d_fixed*1) Fixed effect estimates.
`alpha`	(matrix, n*1) Kernel effect estimates.
`K`	(matrix, n*n) Estimated ensemble kernel matrix.
`u_hat`	(vector of length K) A vector of weights of the kernels in the library.
`kern_term_effect`	(matrix, n*n) Estimated ensemble kernel effect matrix.
`base_est`	(list) The detailed estimation results of K kernels.

Author(s)

Wenying Deng

See Also

strategy: `ensemble`

`generate_kernel` *Generating A Single Kernel*

Description

Generate kernels for the kernel library.

Usage

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

Arguments

`method`	(character) A character string indicating which kernel is to be computed.
`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.
`sigma`	(numeric) The covariance coefficient for neural network kernel.

Details

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

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

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

where $\tilde{x}$ is the vector x prepending with 1.

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_par <- data.frame(method = c("rbf", "polynomial", "matern"),
l = c(.5, 1, 1.5), p = 1:3, sigma = rep(1, 3), stringsAsFactors = FALSE)

kern_func_list <- list()
for (j in 1:nrow(kern_par)) {
  kern_func_list[[j]] <- generate_kernel(kern_par[j, ]$method,
                                         kern_par[j, ]$l,
                                         kern_par[j, ]$p,
                                         kern_par[j, ]$sigma)
}
```

kernel_intercept *Generating A Single Matrix-wise Function Using Intercept*

Description

Generate matrix-wise functions for two matrices using intercept kernel.

Usage

```
kernel_intercept(l, p, sigma)
```

Arguments

`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.
`sigma`	(numeric) The covariance coefficient for neural network kernel.

Details

Polynomial Kernels

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

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

Value

`matrix_wise`	(function) A function calculating the relevance of two matrices.

Author(s)

Wenying Deng

References

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

kernel_linear — *Generating A Single Matrix-wise Function Using Linear*

Description

Generate matrix-wise functions for two matrices using linear kernel.

Usage

```
kernel_linear(l, p, sigma)
```

Arguments

`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.
`sigma`	(numeric) The covariance coefficient for neural network kernel.

Details

Polynomial Kernels

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

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

Value

`matrix_wise`	(function) A function calculating the relevance of two matrices.

Author(s)

Wenying Deng

References

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

kernel_matern *Generating A Single Matrix-wise Function Using Matern*

Description

Generate matrix-wise functions for two matrices using matern kernel.

Usage

```
kernel_matern(l, p, sigma)
```

Arguments

`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.
`sigma`	(numeric) The covariance coefficient for neural network kernel.

Details

Matern Kernels

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

Value

`matrix_wise`	(function) A function calculating the relevance of two matrices.

Author(s)

Wenying Deng

References

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

`kernel_nn` *Generating A Single Matrix-wise Function Using Neural Network*

Description

Generate matrix-wise functions for two matrices using neural network kernel.

Usage

```
kernel_nn(l, p, sigma)
```

Arguments

`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.
`sigma`	(numeric) The covariance coefficient for neural network kernel.

Details

Neural Network Kernels

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

Value

`matrix_wise`	(function) A function calculating the relevance of two matrices.

Author(s)

Wenying Deng

References

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

`kernel_polynomial` *Generating A Single Matrix-wise Function Using Polynomial*

Description

Generate matrix-wise functions for two matrices using polynomial kernel.

Usage

```
kernel_polynomial(l, p, sigma)
```

Arguments

`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.
`sigma`	(numeric) The covariance coefficient for neural network kernel.

Details

Polynomial Kernels

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

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

Value

`matrix_wise`	(function) A function calculating the relevance of two matrices.

Author(s)

Wenying Deng

References

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

`kernel_rational` *Generating A Single Matrix-wise Function Using Rational Quadratic*

Description

Generate matrix-wise functions for two matrices using rational kernel.

Usage

```
kernel_rational(l, p, sigma)
```

Arguments

`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.
`sigma`	(numeric) The covariance coefficient for neural network kernel.

Details

Rational Quadratic Kernels

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

Value

`matrix_wise`	(function) A function calculating the relevance of two matrices.

Author(s)

Wenying Deng

References

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

kernel_rbf *Generating A Single Matrix-wise Function Using RBF*

Description

Generate matrix-wise functions for two matrices using rbf kernel.

Usage

```
kernel_rbf(l, p, sigma)
```

Arguments

`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.
`sigma`	(numeric) The covariance coefficient for neural network kernel.

Details

Gaussian RBF Kernels

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

Value

`matrix_wise`	(function) A function calculating the relevance of two matrices.

Author(s)

Wenying Deng

References

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

`parse_cvek_formula` *Parsing User-supplied Formula*

Description

Parsing user-supplied formula to fixed-effect and kernel matrices.

Usage

```
parse_cvek_formula(
  formula,
  kern_func_list,
  data,
  data_new = NULL,
  verbose = FALSE
)
```

Arguments

`formula`	(formula) A user-supplied formula.
`kern_func_list`	(list) A list of kernel functions in the kernel library
`data`	(data.frame, n*d) A data.frame, list or environment (or object coercible by as.data.frame to a data.frame), containing the variables in formula. Neither a matrix nor an array will be accepted.
`data_new`	(data.frame, n_new*d) New data for computing predictions.
`verbose`	(logical) Whether to print additional messages.

Details

The formula object is exactly like the formula for a GLM except that user can use k() to specify kernel terms. Additionally, user can specify interaction between kernel terms (using either '*' and ':'), and exclude interaction term by including -1 on the RHS of formula.

Value

A list of three slots:

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`K`	(list of matrices) A nested list of kernel term matrices. The first level corresponds to each base kernel function in kern_func_list, the second level corresponds to each kernel term specified in the formula.

Author(s)

Jeremiah Zhe Liu

`predict.cvek` *Predicting New Response*

Description

Predicting new response based on given design matrix and the estimation result.

Usage

```
## S3 method for class 'cvek'
predict(object, newdata, ...)
```

Arguments

`object`	(list) Estimation results returned by cvek() procedure.
`newdata`	(dataframe) The new set of predictors, whose name is the same as those of formula in cvek().
`...`	Further arguments passed to or from other methods.

Details

After we obtain the estimation result, we can predict new response.

Value

`y_pred`	(matrix, n*1) Predicted new response.

Author(s)

Wenying Deng

Examples

```
kern_par <- data.frame(method = rep("rbf", 3),
l = rep(3, 3), p = rep(2, 3),
stringsAsFactors = FALSE)
# define kernel library
kern_func_list <- define_library(kern_par)

n <- 10
d <- 4
formula <- y ~ x1 + x2 + k(x3, x4)
set.seed(1118)
data <- as.data.frame(matrix(
  rnorm(n * d),
  ncol = d,
  dimnames = list(NULL, paste0("x", 1:d))
```

```
))
beta_true <- c(1, .41, 2.37)
lnr_kern_func <- generate_kernel(method = "rbf", l = 3)
kern_effect_lnr <-
  parse_kernel_variable("k(x3, x4)", lnr_kern_func, data)
alpha_lnr_true <- rnorm(n)

data$y <- as.matrix(cbind(1, data[, c("x1", "x2")])) %*% beta_true +
  kern_effect_lnr %*% alpha_lnr_true

data_train <- data[1:6, ]
data_test <- data[7:10, ]

result <- cvek(formula,
               kern_func_list,
               data_train,
               mode = "loocv",
               strategy = "stack",
               beta_exp = 1,
               lambda = exp(seq(-2, 2)),
               test = "asymp",
               alt_kernel_type = "linear",
               verbose = FALSE)

predict(result, data_test)
```

`test_asymp` *Conducting Score Tests for Interaction Using Asymptotic Test*

Description

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

Usage

```
test_asymp(Y, X, y_fixed, alpha0, K_ens, K_int, sigma2_hat, tau_hat, B)
```

Arguments

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`y_fixed`	(vector of length n) Estimated fixed effect of the response.
`alpha0`	(vector of length n) Kernel effect estimator of the estimated ensemble kernel matrix.
`K_ens`	(matrix, n*n) Estimated ensemble kernel matrix.
`K_int`	(matrix, n*n) The kernel matrix to be tested.

`sigma2_hat`	(numeric) The estimated noise of the fixed effect.
`tau_hat`	(numeric) The estimated noise of the kernel effect.
`B`	(integer) A numeric value indicating times of resampling when test = "boot".

Details

Asymptotic Test

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

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 z kova, 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`

`test_boot` *Conducting Score Tests for Interaction Using Bootstrap Test*

Description

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

Usage

```
test_boot(Y, X, y_fixed, alpha0, K_ens, K_int, sigma2_hat, tau_hat, B)
```

Arguments

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`y_fixed`	(vector of length n) Estimated fixed effect of the response.
`alpha0`	(vector of length n) Kernel effect estimator of the estimated ensemble kernel matrix.
`K_ens`	(matrix, n*n) Estimated ensemble kernel matrix.
`K_int`	(matrix, n*n) The kernel matrix to be tested.
`sigma2_hat`	(numeric) The estimated noise of the fixed effect.
`tau_hat`	(numeric) The estimated noise of the kernel effect.
`B`	(integer) A numeric value indicating times of resampling when test = "boot".

Details

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 z kova, 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`

`tuning` *Calculating Tuning Parameters*

Description

Calculate tuning parameters based on given criteria.

Usage

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

Arguments

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`K_mat`	(list of matrices) A nested list of kernel term matrices, corresponding to each kernel term specified in the formula for a base kernel function in kern_func_list.
`mode`	(character) A character string indicating which tuning parameter criteria is to be used.
`lambda`	(numeric) A numeric string specifying the range of tuning parameter to be chosen. The lower limit of lambda must be above 0.

Details

There are seven tuning parameter selections here:

leave-one-out Cross Validation

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

Akaike Information Criteria

$$\lambda_{AIC} = argmin_{\lambda \in \Lambda}\left\{log\ y^{\star T}(I - A_\lambda)^2 y^\star + \frac{2[tr(A_\lambda) + 2]}{n}\right\}$$

Akaike Information Criteria (small-sample variant)

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

Bayesian Information Criteria

$$\lambda_{BIC} = argmin_{\lambda \in \Lambda}\left\{log\ y^{\star T}(I - A_\lambda)^2 y^\star + \frac{log(n)[tr(A_\lambda) + 2]}{n}\right\}$$

Generalized Cross Validation

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

Generalized Cross Validation (small-sample variant)

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

Generalized Maximum Profile Marginal Likelihood

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

Value

`lambda0` (numeric) The selected 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 Principle. 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.

`tuning_AIC` *Calculating Tuning Parameters Using AIC*

Description

Calculate tuning parameters based on AIC.

Usage

```
tuning_AIC(Y, X, K_mat, lambda)
```

Arguments

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`K_mat`	(list of matrices) A nested list of kernel term matrices, corresponding to each kernel term specified in the formula for a base kernel function in kern_func_list.
`lambda`	(numeric) A numeric string specifying the range of tuning parameter to be chosen. The lower limit of lambda must be above 0.

Details

Akaike Information Criteria

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

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 Principle. 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.

tuning_AICc *Calculating Tuning Parameters Using AICc*

Description

Calculate tuning parameters based on AICc.

Usage

```
tuning_AICc(Y, X, K_mat, lambda)
```

Arguments

Y	(matrix, n*1) The vector of response variable.
X	(matrix, n*d_fix) The fixed effect matrix.
K_mat	(list of matrices) A nested list of kernel term matrices, corresponding to each kernel term specified in the formula for a base kernel function in kern_func_list.
lambda	(numeric) A numeric string specifying the range of tuning parameter to be chosen. The lower limit of lambda must be above 0.

Details

Akaike Information Criteria (small sample size)

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

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 Principle. 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.

`tuning_BIC` *Calculating Tuning Parameters Using BIC*

Description

Calculate tuning parameters based on BIC.

Usage

```
tuning_BIC(Y, X, K_mat, lambda)
```

Arguments

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`K_mat`	(list of matrices) A nested list of kernel term matrices, corresponding to each kernel term specified in the formula for a base kernel function in kern_func_list.
`lambda`	(numeric) A numeric string specifying the range of tuning parameter to be chosen. The lower limit of lambda must be above 0.

Details

Bayesian Information Criteria

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

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 Principle. 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.

`tuning_GCV` *Calculating Tuning Parameters Using GCV*

Description

Calculate tuning parameters based on GCV.

Usage

```
tuning_GCV(Y, X, K_mat, lambda)
```

Arguments

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`K_mat`	(list of matrices) A nested list of kernel term matrices, corresponding to each kernel term specified in the formula for a base kernel function in kern_func_list.
`lambda`	(numeric) A numeric string specifying the range of tuning parameter to be chosen. The lower limit of lambda must be above 0.

Details

Generalized Cross Validation

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

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 Principle. 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.

`tuning_GCVc` *Calculating Tuning Parameters Using GCVc*

Description

Calculate tuning parameters based on GCVc.

Usage

```
tuning_GCVc(Y, X, K_mat, lambda)
```

Arguments

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`K_mat`	(list of matrices) A nested list of kernel term matrices, corresponding to each kernel term specified in the formula for a base kernel function in kern_func_list.
`lambda`	(numeric) A numeric string specifying the range of tuning parameter to be chosen. The lower limit of lambda must be above 0.

Details

Generalized Cross Validation (small sample size)

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

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 Principle. 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.

---

tuning_gmpml *Calculating Tuning Parameters Using GMPML*

Description

Calculate tuning parameters based on Generalized Maximum Profile Marginal Likelihood.

Usage

```
tuning_gmpml(Y, X, K_mat, lambda)
```

Arguments

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`K_mat`	(list of matrices) A nested list of kernel term matrices, corresponding to each kernel term specified in the formula for a base kernel function in kern_func_list.
`lambda`	(numeric) A numeric string specifying the range of tuning parameter to be chosen. The lower limit of lambda must be above 0.

Details

Generalized Maximum Profile Marginal Likelihood

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

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 Principle. 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.

`tuning_loocv` *Calculating Tuning Parameters Using looCV*

Description

Calculate tuning parameters based on given leave-one-out Cross Validation.

Usage

```
tuning_loocv(Y, X, K_mat, lambda)
```

Arguments

`Y`	(matrix, n*1) The vector of response variable.
`X`	(matrix, n*d_fix) The fixed effect matrix.
`K_mat`	(list of matrices) A nested list of kernel term matrices, corresponding to each kernel term specified in the formula for a base kernel function in kern_func_list.
`lambda`	(numeric) A numeric string specifying the range of tuning parameter to be chosen. The lower limit of lambda must be above 0.

Details

leave-one-out Cross Validation

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

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 Principle. 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.

Index