Package ‘kernReg’

June 27, 2014

Type Package

Title Linear Principal Components Kernel Regression Methods

Version 1.0

Date 2014-6-8

Author Justin Bleich, Adam Kapelner

Maintainer Justin Bleich <jbleich@wharton.upenn.edu>

Description A tool to select and run kernel PCA linear models for regression and classification

License MIT + file LICENSE

Depends R (>= 3.0), lattice, kernlab, ICEbox, doParallel

R topics documented:

`auto_select_best_kpclr_model` *Auto-Select Best KPCLR Model*

Description

Selects the best KPCLR model based based on finding the minimum cost-weighted error model that has a fn:fp ratio within the bounds defined by `fp_max_cost`, `fn_min_cost`, `fp_min_cost` and `fn_max_cost`. The model can always be hand-selected using `set_desired_model`.

Usage

```
auto_select_best_kpclr_model(explore_kpclr_obj, fp_max_cost = NULL,
  fn_min_cost = NULL, fp_min_cost = NULL, fn_max_cost = NULL)
```

Arguments

`explore_kpclr_obj`
: An object of type `explore_kpclr`.

`fp_max_cost`
: The maximum cost of a false positve. Together with `fn_min_cost`, this will inform the algorithm of the maximum cost ratio of fp:fn. If left to the default `NULL`, the maximum cost ratio will be 25% more than the desired cost ratio.

`fn_min_cost`
: The minimum cost of a false negative. Together with `fp_max_cost`, this will inform the algorithm of the maximum cost ratio of fp:fn. If left to the default `NULL`, the maximum cost ratio will be 25% more than the desired cost ratio.

`fp_min_cost`
: The minimum cost of a false positive. Together with `fn_max_cost`, this will inform the algorithm of the minimum cost ratio of fp:fn. If left to the default `NULL`, the minimum cost ratio will be 25% less than the desired cost ratio.

`fn_max_cost`
: The maximum cost of a false negative. Together with `fp_min_cost`, this will inform the algorithm of the minimum cost ratio of fp:fn. If left to the default `NULL`, the minimum cost ratio will be 25% less than the desired cost ratio.

Value

An object of type `explore_kpclr` with information regarding the auto- selected model.

Author(s)

Adam Kapelner and Justin Bleich

Examples

```
## Not run:
#first create classification data
X = matrix(rnorm(300), ncol = 4)
y = rbinom(300, 1, 0.5)
#now explore kernel models using the default kernel list
explore_kpclr_obj = explore_kpclr_models(X, y)
#now we plot to see how the models built on the training data performed on the validation data
plot(explore_kpclr_obj)
#we are comfortable with allowing the computer to decide which model is best based on
#the minimum cost-weighted error model which falls wthin a default range of the error ratio.
explore_kpclr_obj = auto_select_best_kpclr_model(explore_kpclr_obj)
#re-plotting shows a blue line indicating the favored model
plot(explore_kpclr_obj)

## End(Not run)
```

`auto_select_best_kpcr_model` *Auto-Select Best KPCR Model*

Description

Selects the best KPCR model based on finding the minimum sum of squared error on the validation data. The model can always be hand-selected using `set_desired_model`.

Usage

```
auto_select_best_kpcr_model(explore_kpcr_obj)
```

Arguments

`explore_kpcr_obj`
An object of type `explore_kpcr`.

Value

An object of type `explore_kpcr` with information regarding the auto- selected model.

Author(s)

Adam Kapelner and Justin Bleich

Examples

```
## Not run:
#first create data
X = matrix(rnorm(300), ncol = 4)
y = rnorm(300)
#now explore kernel models using the default kernel list
explore_kpcr_obj = explore_kpcr_models(X, y)
#now we plot to see how the models built on the training data performed on the validation data
plot(explore_kpcr_obj)
```

```
#we are comfortable with allowing the computer to decide which model is best based on lowest SSE
explore_kpcr_obj = auto_select_best_kpcr_model(explore_kpcr_obj)
#re-plotting shows a blue line indicating the favored model
plot(explore_kpcr_obj)

## End(Not run)
```

---

build_final_kpclr_or_kpcr_model
Create Final Kernel Model

Description

Once the user has finished exploring different kernel regressions via explore_kpclr_models or explore_kpcr_models and has estimated future performance on the test data via eval_winning_r_model_on_test_data or eval_winning_lr_model_on_test_data, we now build the final kernel model using all the data from X, y.

Usage

```
build_final_kpclr_or_kpcr_model(explore_kpclr_or_kpcr)
```

Arguments

explore_kpclr_or_kpcr
: The object built from explore_kplr_models or explore_kplcr_models.

Value

The model corresponding to the winning_kernel_num and the winning_rho_num housed in the explore object.

Author(s)

Adam Kapelner and Justin Bleich

Examples

```
## Not run:
#This example is for regression, but it works the same for logistic regression.
#first create regression data
X = matrix(rnorm(300), ncol = 4)
y = rbinom(300, 1, 0.5)
#now explore kernel models using the default kernel list and misclassification costs
explore_kpcr_obj = explore_kpclr_models(X, y)
#now we plot to see how the models built on the training data performed on the validation data
plot(explore_kpcr_obj)
#suppose we choose the 2nd kernel and the 10th rho
explore_kpcr_obj = set_desired_model(explore_kpcr_obj, 2, 10)
#now we build this model using the training and validation data and assess
#out-of-sample performance by predicting on the test data
explore_kpcr_obj = eval_winning_lr_model_on_test_data(explore_kpcr_obj)
#show results to console
explore_kpcr_obj
```

```
#we build a model using all the data in [X, y] to provide to the user who will use
#it to predict on future cases. This model should perform slightly better than the
#out-of-sample test split prediction results printed to console above
model_for_future_prediction = build_final_kpclr_or_kpcr_model(explore_kpcr_obj)

## End(Not run)
```

`build_kpca_object` *Builds a Kernel PCA Object*

Description

Based on the original design matrix (the "data"), build an object which houses information about the data in a transformed / kernelized space based on a kernel of the user's choice. This function will standardize (i.e. center and scale) each predictor column.

Usage

```
build_kpca_object(X, kernel_type, params = c())
```

Arguments

`X`	The original design matrix
`kernel_type`	One of the valid kernel types: vanilla, rbf, poly, tanh, bessel, laplace, anova, spline.
`params`	A list of parameters specific to the kernel of the user's choice. Each kernel type has a required number of parameters that must be passed otherwise the function will throw an error.

Value

A list composed of the original data, the kernel, the K matrix, the centered K matrix, the non-zero eigenvalues and eigenvectors of K and K in the eigenbasis

Author(s)

Justin Bleich and Adam Kapelner

References

Berk, R., Bleich, J., Kapelner, A., Henderson, J. and Kurtz, E., Using Regression Kernels to Forecast A Failure to Appear in Court. (2014) working paper

See Also

`dots`

Examples

```
#create a random predictor matrix with four predictors
X = matrix(rnorm(100), ncol = 4)
#build a KPCA object using the anova kernel with hyperparameters sigma = 0.1 and d = 3
kpca_obj = build_kpca_object(X, "anova", c(0.1, 3))
#display some information to the console
kpca_obj
```

---

eval_winning_lr_model_on_test_data

Evaluate Test Data

Description

After a "satisfactory" model is selected by the user using the `explore_kpcr_models` function, we now predict on the test data to get a glimpse into this model's future out-of-sample performance. Warning: once this is done, you cannot "go back" and "try" to assess performance on new kernels as this would then be snooping. Run this function when you are ready to close the books on this data set and never look back.

Usage

```
eval_winning_lr_model_on_test_data(explore_kpclr_obj,
  use_validation_data = TRUE)
```

Arguments

`explore_kpclr_obj`
: This object is built from `explore_kplcr_models`. We assume the user has updated this object with a satisfactory model by settings `winning_kernel_num` to denote which kernel is selected for the final model and setting `winning_rho_num` to denote which proportion of the variance of the kernel matrix is selected for the final model.

`use_validation_data`
: Should we use the validation data along with the training data. Default is `TRUE`. From our experience, leaving this `FALSE` allows models with better out-of-sample error ratios (number of false negatives to false positives or vice versa). The tradeoff is a larger overall misclassification error because the model is build with the sample size of the training data, not the training plus the validation data.

Value

An expanded `explore_kpclr` list object with new entries that contain information about the performance of the final model on the test data: `test_confusion`, the confusion matrix of the test data; `test_confusion_proportions`, the confusion matrix of the test data as proportions of the number of test observations; `test_misclassification_error`, the total misclassification error of the test data and `test_weighted_cost`, the cost of the errors made as defined by the `fn_cost` and `fp_cost` specified by the user when constructing the model via `explore_kplcr_models`.

Author(s)

Adam Kapelner and Justin Bleich

See Also

explore_kpclr_models

Examples

```
## Not run:
#first create binary classification data
X = matrix(rnorm(300), ncol = 4)
y = rbinom(300, 1, 0.5)
#now explore kernel models using the default kernel list and misclassification costs
explore_kpclr_obj = explore_kpclr_models(X, y)
#now we plot to see how the models built on the training data performed on the validation data
plot(explore_kpclr_obj)
#suppose we choose the 2nd kernel and the 10th rho
explore_kpclr_obj = set_desired_model(explore_kpclr_obj, 2, 10)
#we can re-plot to ensure the chosen model is properly marked with a vertical line
plot(explore_kpclr_obj)
#now we build this model using the training and validation data and assess
#out-of-sample performance by predicting on the test data
explore_kpclr_obj = eval_winning_lr_model_on_test_data(explore_kpclr_obj)
#show results to console
explore_kpclr_obj

## End(Not run)
```

---

eval_winning_r_model_on_test_data
Evaluate Test Data

Description

After a "satisfactory" model is selected by the user using the `explore_kpcr_models` function, we now predict on the test data to get a glimpse into this model's future out-of-sample performance. Warning: once this is done, you cannot "go back" and "try" to assess performance on new kernels as this would then be snooping. Run this function when you are ready to close the books on this data set and never look back.

Usage

```
eval_winning_r_model_on_test_data(explore_kpcr, use_validation_data = TRUE)
```

Arguments

`explore_kpcr` This object is built from `explore_kpcr_models`. We assume the user has updated this object with a satisfactory model by settings `winning_kernel_num` to denote which kernel is selected for the final model and setting `winning_rho_num` to denote which proportion of the variance of the kernel matrix is selected for the final model.

`use_validation_data` Should we use the validation data along with the training data. Default is `TRUE`. From our experience, leaving this `FALSE` allows models with better out-of-sample error ratios (number of false negatives to false positives or vice versa). The tradeoff is a larger overall misclassification error because the model is build with the sample size of the training data, not the training plus the validation data.

Value

An expanded `explore_kpcr` list object with new entries that contain information about the performance of the final model on the test data: `L2_err`, the sum of squared error; `rmse`, the root mean squared error and `L1_err`, the sum of absolute error.

Author(s)

Adam Kapelner and Justin Bleich

See Also

`explore_kpcr_models`

Examples

```
## Not run:
#first create regression data
X = matrix(rnorm(300), ncol = 4)
y = rnorm(300)
#now explore kernel models using the default kernel list
explore_kpcr_obj = explore_kpcr_models(X, y)
#now we plot to see how the models built on the training data performed on the validation data
plot(explore_kpcr_obj)
#suppose we choose the 2nd kernel and the 10th rho
explore_kpcr_obj = set_desired_model(explore_kpcr_obj, 2, 10)
#we can re-plot to ensure the chosen model is properly marked with a vertical line
plot(explore_kpcr_obj)
#now we build this model using the training and validation data and assess
#out-of-sample performance by predicting on the test data
explore_kpcr_obj = eval_winning_r_model_on_test_data(explore_kpcr_obj)
#show results to console
explore_kpcr_obj

## End(Not run)
```

`explore_kpclr_models` *Explore KPCLR for many kernels*

Description

Performs the full procedure outlined in the paper. We first take three splits of the data randomly and a cost ratio. Then, we take in a list of kernels. For each kernel, we build a KPCR model on the training data (the first split). Then we look at performance of each kernel over a range of # of principle components (by proportion of kernel matrix explained, ρ). We pick the proportion that is closest to the predicted cost ratio for each kernel. Now, for each kernel, we have a model. We then predict all models on the validation data (the second split) and pick the best model. Using this best model, we predict on the test data (the third split) and report the out-of-sample statistics

Usage

```
explore_kpclr_models(X, y, kernel_list = NULL, seed = 0,
  split_props = c(1/3, 1/3, 1/3), rho_seq = seq(from = 0.3, to = 0.95, by =
  0.05), fn_cost = 1, fp_cost = 1, family = "binomial", num_cores = 1)
```

Arguments

`X`	The data's predictor matrix.
`y`	The data's response vector.
`kernel_list`	A list of kernels to assess performance of over a variety of PC's by % explained. Each element of this list is a list itself with keys "kernel_type" and "params." If left unspecified, the default are four ANOVA models: (1) sigma = 0.1, d = 2 (2) sigma = 100 d = 2 (3) sigma = 0.1, d = 3 (4) sigma = 100, d = 3
`seed`	A seed to set the random generator. This is required because re-runs will result in different training-validation-test splits which would allow a user to snoop on the test data. We default this to 0 but please use your own seeds to avoid seed conflict.
`split_props`	When splitting the data into training, validation and test sets respectively, what proportions are assigned to each set? This must be numeric of size 3 and the numbers will be normalized to sum to one. The default split is uniform (1/3, 1/3, 1/3).
`rho_seq`	A collection of proportions of the variance explained of the kernel matrix to use. The default is 30%, 35%, ..., 95%.
`fn_cost`	The target cost of a false negative (defaults to 1). Together with `fp_cost`, this will inform the algorithm of the desired ratio of costs.
`fp_cost`	The target cost of a false positive (defaults to 1). Together with `fn_cost`, this will inform the algorithm of the desired cost ratio of fp:fn.
`family`	The family parameter to be passed to the glm function. Default is "binomial." Note that when family is set to "quasibinomial," AIC calculations are not possible.
`num_cores`	The number of cores to use in parallel during computation.

Value

An object of class `explore_kplcr` which is a list housing the results of the procedure

Author(s)

Justin Bleich and Adam Kapelner

References

Berk, R., Bleich, J., Kapelner, A., Henderson, J. and Kurtz, E., Using Regression Kernels to Forecast A Failure to Appear in Court. (2014) working paper

See Also

`kpclr`

`explore_kpcr_models` *Explore KPCR for many kernels*

Description

Performs the full procedure outlined in the paper. We first take three splits of the data randomly and a cost ratio. Then, we take in a list of kernels. For each kernel, we build a KPCR model on the training data (the first split). Then we look at performance of each kernel over a range of # of principle components (by proportion of kernel matrix explained, ρ). We pick the proportion that is closest to the predicted cost ratio for each kernel. Now, for each kernel, we have a model. We then predict all models on the validation data (the second split) and pick the best model. Using this best model, we predict on the test data (the third split) and report the out-of-sample statistics

Usage

```
explore_kpcr_models(X, y, kernel_list = NULL, seed = 0,
  split_props = c(1/3, 1/3, 1/3), rho_seq = seq(from = 0.3, to = 0.95, by =
  0.05), num_cores = 1)
```

Arguments

`X`	The data's predictor matrix
`y`	The data's response vector
`kernel_list`	A list of kernels to assess performance of over a variety of PC's by % explained. Each element of this list is a list itself with keys "kernel_type" and "params." If left unspecified, the default are four ANOVA models: (1) sigma = 0.1, d = 2 (2) sigma = 100 d = 2 (3) sigma = 0.1, d = 3 (4) sigma = 100, d = 3
`seed`	A seed to set the random generator. This is required because re-runs will result in different training-validation-test splits which would allow a user to snoop on the test data. We default this to 0 but please use your own seeds to avoid seed conflict.
`split_props`	When splitting the data into training, validation and test sets respectively, what proportions are assigned to each set? This must be numeric of size 3 and the numbers will be normalized to sum to one. The default split is uniform (1/3, 1/3, 1/3).
`rho_seq`	A collection of proportions of the variance explained of the kernel matrix to use. The default is 30%, 35%, ..., 95%.
`num_cores`	The number of cores to use in parallel during computation.

Value

An object of class `explore_kplcr` which is a list housing the results of the procedure.

Author(s)

Justin Bleich and Adam Kapelner

References

Berk, R., Bleich, J., Kapelner, A., Henderson, J. and Kurtz, E., Using Regression Kernels to Forecast A Failure to Appear in Court. (2014) working paper

See Also

kpcr

---

kernReg — *Linear Principal Components Kernel Regression Methods*

Description

A tool to select and run kernel PCA linear models for regression and classification

Author(s)

Justin Bleich <bleich@wharton.upenn.edu> and Adam Kapelner <kapelner@wharton.upenn.edu>

References

Berk, R., Bleich, J., Kapelner, A., Henderson, J. and Kurtz, E., Using Regression Kernels to Forecast A Failure to Appear in Court. (2014) working paper

---

kpclr — *Run a logistic regression using Kernel PCA*

Description

`kpclr` runs a logistic regression on features created from an eigendecomposition of a certain dimension of the kernelized data

Usage

```
kpclr(kpca_object, y, num_pcs = NULL, frac_var = NULL, weights = NULL,
  family = "binomial")
```

Arguments

`kpca_object`	The object that contains the kernel and the kernelized data with its eigendecomposition
`y`	The response to be regressed on the features which are the principal components of the kernelized data
`num_pcs`	The number of principal components to use for the regression (this or `frac_var` must be specified)
`frac_var`	Pick the number of principal components to use based on the fraction of variance to explain (this or `num_pcs` must be specified)
`weights`	Weights to be used on each observation in a weighted generalized least squares implementation. If not specified (default), uniform weights are used
`family`	The family parameter to be passed to the glm function. Default is "binomial." Note that with this default, AIC calculations are not possible.

Value

An lm object with the kpca_object embedded as well as the number of principal components used

Author(s)

Justin Bleich and Adam Kapelner

References

Berk, R., Bleich, J., Kapelner, A., Henderson, J. and Kurtz, E., Using Regression Kernels to Forecast A Failure to Appear in Court. (2014) working paper

See Also

`kpcr`

---

kpcr — *Run a linear regression using Kernel PCA*

Description

`kpcr` runs a linear regression on features created from an eigendecomposition of a certain dimension of the kernelized data

Usage

```
kpcr(kpca_object, y, num_pcs = NULL, frac_var = NULL)
```

Arguments

`kpca_object`	The object that contains the kernel, the kernelized data with its eigendecomposition
`y`	The response to be regressed on the features which are the principal components of the kernelized data
`num_pcs`	The number of principal components to use for the regression (this or `frac_var` must be specified)
`frac_var`	Pick the number of principal components to use based on the fraction of variance to explain (this or `num_pcs` must be specified)

Value

An lm object with the kpca_object embedded as well as the number of principal components used

Author(s)

Justin Bleich and Adam Kapelner

References

Berk, R., Bleich, J., Kapelner, A., Henderson, J. and Kurtz, E., Using Regression Kernels to Forecast A Failure to Appear in Court. (2014) working paper

See Also

kpclr

`plot.explore_kpclr` *Explore kpclr plot*

Description

This is an S3 convenience method for the function plot_explore_kpclr. Please follow the link for the full documentation.

Usage

```
## S3 method for class explore_kpclr
plot(x, ...)
```

Arguments

x	The `explore_kpclr` object to plot
...	Other parameters to pass to plot_explore_kpclr.

Author(s)

Adam Kapelner and Justin Bleich

`plot.explore_kpcr` *Explore KPCR plot*

Description

This is an S3 convenience method for the function plot_explore_kpcr. Please follow the link for the full documentation.

Usage

```
## S3 method for class explore_kpcr
plot(x, ...)
```

Arguments

x	The `explore_kpcr` object to plot
...	Other parameters to pass to plot_explore_kpcr.

Author(s)

Adam Kapelner and Justin Bleich

plot.kpca *Plots the kernel matrix*

Description

This is an S3 convenience method for the function `plot_kpca`. Please follow the link for the full documentation.

Usage

```
## S3 method for class kpca
plot(x, ...)
```

Arguments

`x`	The kpca object to plot
`...`	Other parameters to pass to `plot_kpca`.

Author(s)

Adam Kapelner and Justin Bleich

plot_explore_kpclr *Explore kpclr plot*

Description

Many kernel principle components logistic regression models were fitted on the training data via `explore_kpclr_models`. This function will create one plot for each of the kernel models investigated. At every value of rho (the proportion of variance of the kernel matrix explained), the following three things will be plotted (1) the ratio of the number of false negatives to the number of false positives when predicted out of sample (2) the AIC of the model (3) the cost-weighted error of the out-of-sample validation data.

Usage

```
plot_explore_kpclr(explore_kpclr_obj, tile_cols = 3, ylim = NULL,
  min_fn_fp_ratio = NULL, max_fn_fp_ratio = NULL,
  quantile_aic_to_display = 0.75, quantile_cwe_to_display = 0.95,
  color_winning_model = "blue", color_num_fn_fp_ratio = "black",
  color_aic = "firebrick3", color_cwe = "forestgreen",
  show_rho_numbers = TRUE, text_label_offset_pct = 0.1,
  kernels_to_plot = NULL, ...)
```

Arguments

explore_kpclr_obj
: An object of type explore_kpclr built with explore_kpclr_models

tile_cols
: When plotting all kernel model performances, how many kernels per plot window column? Default is 3.

ylim
: The ylim parameter which is passed to the plot function.

min_fn_fp_ratio
: If specified, plots a horizontal line on the y-axis representing the lower bound of the ratio of the number of false negatives to false positives. Defaults to the value set by auto_select_best_kpclr_model (if it was run previously). If not, no line is plotted.

max_fn_fp_ratio
: If specified, plots a horizontal line on the y-axis representing the upper bound of the ratio of the number of false negatives to false positives. Defaults to the value set by auto_select_best_kpclr_model (if it was run previously). If not, no line is plotted.

quantile_aic_to_display
: When plotting the AICs for each model, which quantile should be truncated? Default is 95%.

quantile_cwe_to_display
: When plotting the cost-weighted-errors for each model, which quantile should be truncated? Default is 75%.

color_winning_model
: What color is the vertical line of the winning model. Default is blue.

color_num_fn_fp_ratio
: What color are the ratios of false negatives to false positive? The default is black.

color_aic
: What color are the AIC lines? Default is reddish.

color_cwe
: What color are the cost weighted error lines? Default is greenish.

show_rho_numbers
: Plot the rho number on each of the exploratory plots. Default is TRUE.

text_label_offset_pct
: If the rho numbers are plotted, what percent offset below the points? Default is 10%.

kernels_to_plot
: A list of indices of the kernels to plot. If left to the default NULL, all kernels are plotted.

...
: Other parameters to pass to plot. Of particular interest is xlim which will limit some models from being displayed.

Author(s)

Adam Kapelner and Justin Bleich

plot_explore_kpcr *Explore KPCR plot*

Description

Many kernel principle components logistic regression models were fitted on the training data via explore_kpcr_models. This function will create one plot for each of the kernel models investigated. At every value of rho (the proportion of variance of the kernel matrix explained), the AIC of the model and the sse of the out-of-sample validation data is plotted.

Usage

```
plot_explore_kpcr(explore_kpcr_obj, tile_cols = 3,
  quantile_aic_to_display = 0.75, color_winning_model = "blue",
  color_sse = "black", color_aic = "firebrick3", show_rho_numbers = TRUE,
  text_label_offset_pct = 0.1, label_skip = 3, ...)
```

Arguments

- `explore_kpcr_obj` An object of type `explore_kpcr` built with explore_kpcr_models
- `tile_cols` When plotting all kernel model performances, how many kernels per plot window column? Default is `3`.
- `quantile_aic_to_display` When plotting the AICs for each model, which quantile should be truncated? Default is `75%`.
- `color_winning_model` What color is the vertical line of the winning model. Default is blue.
- `color_aic` What color are the AICs? Default is reddish.
- `color_sse` What color are the SSEs? Default is greenish.
- `show_rho_numbers` Plot the rho number on each of the exploratory plots. Default is `TRUE`.
- `text_label_offset_pct` If the rho numbers are plotted, what percent offset below the points? Default is `10%`.
- `label_skip` If the rho numbers are plotted, how many should be skipped? Default is `3`.
- `...` Other parameters to pass to plot.

Author(s)

Adam Kapelner and Justin Bleich

plot_kpca *Plots the kernel matrix*

Description

Illustrates the kernel matrix as a heatmap.

Usage

```
plot_kpca(kpca_object, lower_triangular = TRUE, transform = NULL,
  col.regions = rainbow(200, end = 0.78), main = NULL, ...)
```

Arguments

- `kpca_object` The kpca object to plot
- `lower_triangular` If the kernel is symmetric (the usual case), setting this to `TRUE` will only plot the lower triangle. Default is `TRUE`.
- `col.regions` How to color the heatmap. The default is color but if you would like to produce a grayscale images for publication, you can use something like `gray(100 : 0 / 100)`.
- `transform` An optional function to transform the entries of `K`.
- `main` Title of the plot. Default is the description of the kernel.
- `...` Other parameters to pass to levelplot. Especially useful are `xlim` and `ylim` to look at a portion of the matrix. Also, use the `at` to mimic limits on the magnitudes of the entries of K. For instance `seq(1000, 2000, by = 20)` will only graph entries between 1,000 and 2,000. Do not change the "by" parameter here, 20 seems to be a good choice. Standardize the `at` parameter allows better apples : apples comparisons across the same kernel with different hyperparameters (e.g. the radial basis kernel with different gamma values).

Author(s)

Adam Kapelner and Justin Bleich

Examples

```
#create a random predictor matrix with four predictors
X = matrix(rnorm(100), ncol = 4)
#build a KPCA object using the anova kernel with hyperparameters sigma = 0.1 and d = 3
kpca_obj = build_kpca_object(X, "anova", c(0.1, 3))
#visualize the kernel
plot_kpca(kpca_obj) #"plot(kpca_obj)" also works and is recommended
```

`plot_pdp` *Plots a PDP for a kernel regression*

Description

`plot_pdp` plots a partial dependence plot (PDP; Friedman, 2001) for one of the predictors. The function allows the PDP to be plotted to arbitrary accuracy and allows parameters to be passed for customization of the plotting. This function relies on package **ICEbox**.

Usage

```
plot_pdp(kpca_model, predictor, type = "link", frac_to_build = 1, ...)
```

Arguments

`kpca_model`	A linear or logistic kernel PCA regression model
`predictor`	The predictor to build a PDP for. It can be the number of the column or the column's name
`type`	The type argument that gets passed to the `kpca_predict` function
`frac_to_build`	The fraction of the design matrix to construct the PDP from.
`...`	Other parameters to pass to the `plot` function

Author(s)

Justin Bleich and Adam Kapelner

References

Berk, R., Bleich, J., Kapelner, A., Henderson, J. and Kurtz, E., Using Regression Kernels to Forecast A Failure to Appear in Court. (2014) working paper

See Also

`predict.kpcr`, `predict.kpclr`, `plot`

`predict.kpclr` *Predicts for new data*

Description

`predict.kpclr` predicts using the kernel PCA logistic model for new data

Usage

```
## S3 method for class kpclr
predict(object, new_data, type = "response", num_cores = 1,
  ...)
```

Arguments

`object`	The Kernel PCA logistic model used to predict
`new_data`	The new data the user wishes to predict
`type`	Which output to return to the user. Use "response" for predicted probability and "link" for a logit (see `predict.glm` for more information)
`num_cores`	Number of cores for parallel prediction
`...`	Other parameters to be passed to `predict.glm`

Value

A vector of predictions with lenth of the number of rows of `new_data` generated via `predict.glm`

Author(s)

Justin Bleich and Adam Kapelner

See Also

`predict.glm`

Examples

```
## Not run:
#first create binary classification data
X = matrix(rnorm(300), ncol = 4)
y = rbinom(300, 1, 0.5)
#build a KPCA object using the anova kernel with hyperparameters sigma = 0.1 and d = 3
kpca_obj = build_kpca_object(X, "anova", c(0.1, 3))
#build a kpclr model using 75% of the variance in the kernel matrix and weights for 1:1 cost ratio
kpclr_mod = kpclr(kpca_obj, y, frac_var = 0.75, weights = weights_for_kpclr(y))
#create 10 new data records and forecast on the new data
x_star = matrix(rnorm(40), ncol = 4)
y_hat = predict(kpclr_mod, x_star)

## End(Not run)
```

`predict.kpcr` *Predicts for new data*

Description

`predict.kpcr` predicts using the kernel PCA model for new data

Usage

```
## S3 method for class kpcr
predict(object, new_data, num_cores = 1, ...)
```

Arguments

`object`	The Kernel PCA linear model object used to predict
`new_data`	The new data the user wishes to predict
`num_cores`	Number of cores for parallel prediction
`...`	Other parameters to be passed to `predict.lm`

Value

A vector of predictions with lenth of the number of rows of `new_data` generated via `predict.lm`

Author(s)

Justin Bleich and Adam Kapelner

See Also

`predict.lm`

Examples

```
## Not run:
#first create regression data
X = matrix(rnorm(300), ncol = 4)
y = rnorm(300)
#build a KPCA object using the anova kernel with hyperparameters sigma = 0.1 and d = 3
kpca_obj = build_kpca_object(X, "anova", c(0.1, 3))
#build a kpcr model using 75% of the variance in the kernel matrix
kpcr_mod = kpclr(kpca_obj, y, frac_var = 0.75)
#create 10 new data records and forecast on the new data
x_star = matrix(rnorm(40), ncol = 4)
y_hat = predict(kpcr_mod, x_star)

## End(Not run)
```

`print.explore_kpclr` *Prints a summary of a explore_kpclr object*

Description

Prints a summary of a explore_kpclr object

Usage

```
## S3 method for class explore_kpclr
print(x, ...)
```

Arguments

`x`	The explore_kpclr object to be summarized in the console
`...`	Other parameters to pass to the default print function

Author(s)

Justin Bleich and Adam Kapelner

---

print.explore_kpcr — *Prints a summary of a explore_kpcr object*

Description

Prints a summary of a explore_kpcr object

Usage

```
## S3 method for class explore_kpcr
print(x, ...)
```

Arguments

x	The `explore_kpcr` object to be summarized in the console
...	Other parameters to pass to the default print function

Author(s)

Justin Bleich and Adam Kapelner

---

print.kpca — *Prints a summary of a kpca object*

Description

Prints a summary of a kpca object

Usage

```
## S3 method for class kpca
print(x, ...)
```

Arguments

x	The kpca object to be summarized in the console
...	Other parameters to pass to the default print function

Author(s)

Justin Bleich and Adam Kapelner

`print.kpclr` *Prints a summary of a kpclr object*

Description

Prints a summary of a kpclr object

Usage

```
## S3 method for class kpclr
print(x, ...)
```

Arguments

`x`	The kpclr object to be summarized in the console
`...`	Other parameters to pass to the default print function

Author(s)

Justin Bleich and Adam Kapelner

`print.kpcr` *Prints a summary of a kpcr object*

Description

Prints a summary of a kpcr object

Usage

```
## S3 method for class kpcr
print(x, ...)
```

Arguments

`x`	The kpcr object to be summarized in the console
`...`	Other parameters to pass to the default print function

Author(s)

Justin Bleich and Adam Kapelner

`set_desired_model` *Sets Desired Model*

Description

Given an object created with `explore_kpcr_models` or `explore_kpclr_models` and plotted with `plot.explore_kpcr` or `plot.explore_kpclr` respectively, the user now chooses a desired model by selecting a kernel and a rho index from this visualization. This function simply inputs this information into the object returned by the `explore_kpclr_models` or `explore_kpclr_models` functions. If the user wishes to reset the desired model, run this function with `NULL` for the arguments `winning_kernel_num` and `winning_rho_num`.

Usage

```
set_desired_model(explore_kpcr_or_kpclr_obj, winning_kernel_num,
  winning_rho_num)
```

Arguments

`explore_kpcr_or_kpclr_obj`
: The object created by running `explore_kpcr_models` or `explore_kpclr_models`

`winning_kernel_num`
: The desired model's kernel index.

`winning_rho_num`
: The desired model's rho index.

Value

An object of type `exlore_kpcr` or `explore_kpclr` augmented with information about the user's desired model.

Author(s)

Adam Kapelner and Justin Bleich

Examples

```
## Not run:
#Note this is example is for classification, but it works the same for regression
#first create classification data
X = matrix(rnorm(300), ncol = 4)
y = rbinom(300, 1, 0.5)
#now explore kernel models using the default kernel list
explore_kpclr_obj = explore_kpclr_models(X, y)
#now we plot to see how the models built on the training data performed on the validation data
plot(explore_kpclr_obj)
#we believe that the third kernel and the 9th value of rho is the "best" model
explore_kpclr_obj = set_desired_model(explore_kpclr_obj,
winning_kernel_num = 3, winning_rho_num = 9)
#re-plotting shows a blue line indicating the model we just set
plot(explore_kpclr_obj)

## End(Not run)
```

summary.explore_kpclr *Prints a summary of a explore_kpclr object*

Description

Prints a summary of a explore_kpclr object

Usage

```
## S3 method for class explore_kpclr
summary(object, ...)
```

Arguments

`object`	The explore_kpclr object to be summarized in the console
`...`	Other parameters to pass to the default summary function

Author(s)

Justin Bleich and Adam Kapelner

summary.explore_kpcr *Prints a summary of a explore_kpcr object*

Description

Prints a summary of a explore_kpcr object

Usage

```
## S3 method for class explore_kpcr
summary(object, ...)
```

Arguments

`object`	The `explore_kpcr` object to be summarized in the console
`...`	Other parameters to pass to the default summary function

Author(s)

Justin Bleich and Adam Kapelner

summary.kpca *Prints a summary of a kpca object*

Description

Prints a summary of a kpca object

Usage

```
## S3 method for class kpca
summary(object, ...)
```

Arguments

object	The kpca object to be summarized in the console
...	Other parameters to pass to the default summary function

Author(s)

Justin Bleich and Adam Kapelner

summary.kpclr *Prints a summary of a kpclr object*

Description

Prints a summary of a kpclr object

Usage

```
## S3 method for class kpclr
summary(object, ...)
```

Arguments

object	The kpclr object to be summarized in the console
...	Other parameters to pass to the default summary function

Author(s)

Justin Bleich and Adam Kapelner

summary.kpcr *Prints a summary of a kpcr object*

Description

Prints a summary of a kpcr object

Usage

```
## S3 method for class kpcr
summary(object, ...)
```

Arguments

`object`	The kpcr object to be summarized in the console
`...`	Other parameters to pass to the default summary function

Author(s)

Justin Bleich and Adam Kapelner

weights_for_kpclr *Creates weights for kpca logistic regression*

Description

`weights_for_kpca_logistic_regression` will set individual weights for each observation in the training data based on two things (1) the ratio of 1's and 0's in the training set and (2) the user-set false-negative to false-positive ratio.

Usage

```
weights_for_kpclr(y_train, fn_to_fp_ratio = 1)
```

Arguments

`y_train`	The responses for the training data - a binary vector of length `n`.
`fn_to_fp_ratio`	The ratio of the "severity" of the false negative to the "severity" of the false positive (defaults to 1)

Value

A vector of weights of length `n`.

Author(s)

Justin Bleich and Adam Kapelner

Examples

```
y_train = c(rep(0, 100), rep(1, 200))
weights = weights_for_kpclr(y_train)
table(weights)
```

Index