Package 'aum'

June 20, 2024

Type Package

Title Area Under Minimum of False Positives and Negatives

Version 2024.6.19

Description Efficient algorithms
<https://jmlr.org/papers/v24/21-0751.html>
for computing Area Under Minimum,
directional derivatives, and
line search optimization of a linear model,
with objective defined as either
max Area Under the Curve or
min Area Under Minimum.

License GPL-3

LinkingTo Rcpp

URL https://github.com/tdhock/aum

BugReports https://github.com/tdhock/aum/issues

Imports Rcpp, data.table

Suggests testthat, kernlab, nc, ggplot2, WeightedROC, penaltyLearning, knitr, markdown, mlbench, directlabels, microbenchmark, covr, atime, ggrepel

VignetteBuilder knitr

NeedsCompilation yes

Author Toby Dylan Hocking [aut, cre],
Jadon Fowler [aut] (Contributed exact line search C++ code)

Maintainer Toby Dylan Hocking <toby.hocking@r-project.org>

Repository CRAN

Date/Publication 2024-06-20 21:20:02 UTC

Contents

`aum` *aum*

Description

Compute the Area Under Minimum of False Positives and False Negatives, and its directional derivatives.

Usage

```
aum(error.diff.df, pred.vec)
```

Arguments

`error.diff.df`	data frame of error differences, typically computed via `aum_diffs_binary` or `aum_diffs_penalty`. There should be one row for each change in error functions. "example" column indicates example ID (int from 1 to N), "pred" column indicates predicted value where there is a change in the error function(s), "fp_diff" and "fn_diff" columns indicate differences in false positives and false negatives at that predicted value. Note that this representation assumes that each error function has fp=0 at pred=-Inf and fn=0 at pred=Inf.
`pred.vec`	numeric vector of N predicted values.

Value

Named list of two items: aum is numeric scalar loss value, derivative_mat is N x 2 matrix of directional derivatives (first column is derivative from left, second column is derivative from right). If

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

Examples

```
(bin.diffs <- aum::aum_diffs_binary(c(0,1)))
aum::aum(bin.diffs, c(-10,10))
aum::aum(bin.diffs, c(0,0))
aum::aum(bin.diffs, c(10,-10))
```

`aum_diffs` *aum diffs*

Description

Create error differences data table which can be used as input to `aum` function. Typical users should not use this function directly, and instead use `aum_diffs_binary` for binary classification, and `aum_diffs_penalty` for error defined as a function of non-negative penalty.

Usage

```
aum_diffs(example, pred,
    fp_diff, fn_diff,
    pred.name.vec)
```

Arguments

`example`	Integer or character vector identifying different examples.
`pred`	Numeric vector of predicted values at which the error changes.
`fp_diff`	Numeric vector of difference in fp at `pred`.
`fn_diff`	Numeric vector of difference in fn at `pred`.
`pred.name.vec`	Character vector of `example` names for predictions.

Value

data table of class "aum_diffs" in which each rows represents a breakpoint in an error function. Columns are interpreted as follows: there is a change of "fp_diff","fn_diff" at predicted value "pred" for example/observation "example". This can be used for computing Area Under Minimum via `aum` function, and plotted via `plot.aum_diffs`.

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

Examples

```
aum::aum_diffs_binary(c(0,1))
aum::aum_diffs(c("positive", "negative"), 0, c(0,1), c(-1,1), c("negative", "positive"))
rbind(aum::aum_diffs(0L, 0, 1, 0), aum_diffs(1L, 0, 0, -1))
```

`aum_diffs_binary` *aum diffs binary*

Description

Convert binary labels to error differences.

Usage

```
aum_diffs_binary(label.vec,
    pred.name.vec, denominator = "count")
```

Arguments

`label.vec`	Numeric vector representing binary labels (either all 0,1 or all -1,1). If named, names are used to identify each example.
`pred.name.vec`	Character vector of prediction example names, used to convert names of `label.vec` to integers.
`denominator`	Type of diffs, either "count" or "rate".

Value

data table of class "aum_diffs" in which each rows represents a breakpoint in an error function. Columns are interpreted as follows: there is a change of "fp_diff","fn_diff" at predicted value "pred" for example/observation "example". This can be used for computing Area Under Minimum via `aum` function, and plotted via `plot.aum_diffs`.

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

Examples

```
aum_diffs_binary(c(0,1))
aum_diffs_binary(c(-1,1))
aum_diffs_binary(c(a=0,b=1,c=0), pred.name.vec=c("c","b"))
aum_diffs_binary(c(0,0,1,1,1), denominator="rate")
```

aum_diffs_penalty *aum diffs penalty*

Description

Convert penalized errors to error differences. A typical use case is for penalized optimal change-point models, for which small penalty values result in large fp/fn, and large penalty values result in small fp/fn.

Usage

```
aum_diffs_penalty(errors.df,
    pred.name.vec, denominator = "count")
```

Arguments

errors.df	data.frame which describes error as a function of penalty/lambda, with at least columns example, min.lambda, fp, fn. Interpreted as follows: fp/fn occur from all penalties from min.lambda to the next value of min.lambda within the current value of example.
pred.name.vec	Character vector of prediction example names, used to convert names of label.vec to integers.
denominator	Type of diffs, either "count" or "rate".

Value

data table of class "aum_diffs" in which each rows represents a breakpoint in an error function. Columns are interpreted as follows: there is a change of "fp_diff","fn_diff" at predicted value "pred" for example/observation "example". This can be used for computing Area Under Minimum via aum function, and plotted via plot.aum_diffs.

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

Examples

```
if(require("data.table"))setDTthreads(1L)#for CRAN check.

## Simple synthetic example with two changes in error function.
simple.df <- data.frame(
  example=1L,
  min.lambda=c(0, exp(1), exp(2), exp(3)),
  fp=c(6,2,2,0),
  fn=c(0,1,1,5))
(simple.diffs <- aum::aum_diffs_penalty(simple.df))
if(requireNamespace("ggplot2"))plot(simple.diffs)
```

```
(simple.rates <- aum::aum_diffs_penalty(simple.df, denominator="rate"))
if(requireNamespace("ggplot2"))plot(simple.rates)

## Simple real data with four example, one has non-monotonic fn.
if(requireNamespace("penaltyLearning")){
  data(neuroblastomaProcessed, package="penaltyLearning", envir=environment())
  ## assume min.lambda, max.lambda columns only? use names?
  nb.err <- with(neuroblastomaProcessed$errors, data.frame(
    example=paste0(profile.id, ".", chromosome),
    min.lambda,
    max.lambda,
    fp, fn))
  (nb.diffs <- aum::aum_diffs_penalty(nb.err, c("1.2", "1.1", "4.1", "4.2")))
  if(requireNamespace("ggplot2"))plot(nb.diffs)
}

## More complex real data example
data(fn.not.zero, package="aum", envir=environment())
pred.names <- unique(fn.not.zero$example)
(fn.not.zero.diffs <- aum::aum_diffs_penalty(fn.not.zero, pred.names))
if(requireNamespace("ggplot2"))plot(fn.not.zero.diffs)

if(require("ggplot2")){
  name2id <- structure(seq(0, length(pred.names)-1L), names=pred.names)
 fn.not.zero.wide <- fn.not.zero[, .(example=name2id[example], min.lambda, max.lambda, fp, fn)]
  fn.not.zero.tall <- data.table::melt(fn.not.zero.wide, measure=c("fp", "fn"))
  ggplot()+
    geom_segment(aes(
      -log(min.lambda), value,
      xend=-log(max.lambda), yend=value,
      color=variable, linewidth=variable),
      data=fn.not.zero.tall)+
    geom_point(aes(
      -log(min.lambda), value,
      fill=variable),
      color="black",
      shape=21,
      data=fn.not.zero.tall)+
    geom_vline(aes(
      xintercept=pred),
      data=fn.not.zero.diffs)+
    scale_size_manual(values=c(fp=2, fn=1))+
    facet_grid(example ~ ., labeller=label_both)
}
```

`aum_errors` *aum errors*

Description

Convert diffs to canonical errors, used internally in `plot.aum_diffs`.

Usage

```
aum_errors(diffs.df)
```

Arguments

diffs.df	data.table of diffs from aum_diffs.

Value

data.table suitable for plotting piecewise constant error functions, with columns example, min.pred, max.pred, fp, fn.

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

Examples

```
(bin.diffs <- aum::aum_diffs_binary(c(0,1)))
if(requireNamespace("ggplot2"))plot(bin.diffs)
aum::aum_errors(bin.diffs)
```

---

aum_linear_model *aum linear model*

Description

Learn a linear model with weights that minimize AUM. Weights are initialized as a vector of zeros, then optimized using gradient descent with exact line search.

Usage

```
aum_linear_model(feature.list,
    diff.list, max.steps = NULL,
    improvement.thresh = NULL,
    maxIterations = "min.aum",
    initial.weight.fun = NULL,
    line.search.set = "subtrain")
```

Arguments

feature.list	List with named elements subtrain and optionally validation, each should be a scaled feature matrix.
diff.list	List with named elements subtrain and optionally validation, each should be a data table of differences in error functions.

`max.steps` positive integer: max number of steps of gradient descent with exact line search (specify either this or `improvement.thresh`, not both).

`improvement.thresh` non-negative real number: keep doing gradient descent while the improvement in objective is greater than this number (specify either this or `max.steps`, not both).

`maxIterations` max number of iterations of exact line search. If "max.auc" then the objective for `improvement.thresh` is max AUC, otherwise objective is min AUM. Default is "min.aum"

`initial.weight.fun` Function for computing initial weights, default NULL means use a random standard normal vector.

`line.search.set` set to use for line search, subtrain or validation.

Value

Linear model represented as a list of class aum_linear_model with named elements: loss is a data table of values for subtrain and optionally validation at each step, weight.vec is the final vector of weights learned via gradient descent, intercept is the value which results in minimal total error (FP+FN), learned via a linear scan over all possible values given the final weight vector, and search is a data table with one row for each step (best step size and number of iterations of line search).

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

`aum_linear_model_cv` *aum linear model cv*

Description

Cross-validation for learning number of early stopping gradient descent steps with exact line search, in linear model for minimizing AUM.

Usage

```
aum_linear_model_cv(feature.mat,
    diff.dt, maxIterations = "min.aum",
    improvement.thresh = NULL,
    n.folds = 3, initial.weight.fun = NULL)
```

Arguments

feature.mat	N x P matrix of features, which will be scaled before gradient descent.
diff.dt	data table of differences in error functions, from aum_diffs_penalty or aum_diffs_binary. There should be an example column with values from 0 to N-1.
maxIterations	max iterations of the exact line search, default is number of examples.
improvement.thresh	before doing cross-validation to learn the number of gradient descent steps, we do gradient descent on the full data set in order to determine a max number of steps, by continuing to do exact line search steps while the decrease in AUM is greater than this value (positive real number). Default NULL means to use the value which is ten times smaller than the min non-zero absolute value of FP and FN diffs in diff.dt.
n.folds	Number of cross-validation folds to average over to determine the best number of steps of gradient descent.
initial.weight.fun	Function for computing initial weight vector in gradient descent.

Value

Model trained with best number of iterations, represented as a list of class aum_linear_model_cv with named elements: keep is a logical vector telling which features should be kept before doing matrix multiply of learned weight vector, weight.orig/weight.vec and intercept.orig/intercept are the learned weights/intercepts for the original/scaled feature space, fold.loss/set.loss are data tables of loss values for the subtrain/validation sets, used for selecting the best number of gradient descent steps.

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

Examples

```
if(require("data.table"))setDTthreads(1L)#for CRAN check.

## simulated binary classification problem.
N.rows <- 60
N.cols <- 2
set.seed(1)
feature.mat <- matrix(rnorm(N.rows*N.cols), N.rows, N.cols)
unknown.score <- feature.mat[,1]*2.1 + rnorm(N.rows)
label.vec <- ifelse(unknown.score > 0, 1, 0)
diffs.dt <- aum::aum_diffs_binary(label.vec)

## Default line search keeps doing iterations until increase in AUM.
(default.time <- system.time({
  default.model <- aum::aum_linear_model_cv(feature.mat, diffs.dt)
}))
plot(default.model)
```

```
print(default.valid <- default.model[["set.loss"]][set=="validation"])
print(default.model[["search"]][, .(step.size, aum, iterations=q.size)])

## Can specify max number of iterations of line search.
(small.step.time <- system.time({
 small.step.model <- aum::aum_linear_model_cv(feature.mat, diffs.dt, maxIterations = N.rows)
}))
plot(small.step.model)
print(small.step.valid <- small.step.model[["set.loss"]][set=="validation"])
small.step.model[["search"]][, .(step.size, aum, iterations=q.size)]

## Compare number of steps, iterations and time. On my machine small
## step model takes more time/steps, but less iterations in the C++
## line search code.
cbind(
  iterations=c(
    default=default.model[["search"]][, sum(q.size)],
    small.step=small.step.model[["search"]][, sum(q.size)]),
  seconds=c(
    default.time[["elapsed"]],
    small.step.time[["elapsed"]]),
  steps=c(
    default.model[["min.valid.aum"]][["step.number"]],
    small.step.model[["min.valid.aum"]][["step.number"]]),
  min.valid.aum=c(
    default.model[["min.valid.aum"]][["aum_mean"]],
    small.step.model[["min.valid.aum"]][["aum_mean"]]))
```

`aum_linear_model_ls` *aum linear model ls*

Description

Learn a linear model with weights that minimize AUM. Weights are initialized as a vector of zeros, then optimized using gradient descent with exact line search.

Usage

```
aum_linear_model_ls(feature.list,
    diff.list, max.steps = NULL,
    improvement.thresh = NULL,
    maxIterations = "min.aum",
    initial.weight.fun = NULL)
```

Arguments

`feature.list`	List with named elements subtrain and validation, each should be a scaled feature matrix.

`diff.list` List with named elements subtrain and validation, each should be a data table of differences in error functions.

`max.steps` positive integer: max number of steps of gradient descent with exact line search (specify either this or `improvement.thresh`, not both).

`improvement.thresh` non-negative real number: keep doing gradient descent while the improvement in objective is greater than this number (specify either this or `max.steps`, not both).

`maxIterations` max number of iterations of exact line search. If "max.auc" then the objective for `improvement.thresh` is max AUC, otherwise objective is min AUM. Default is "min.aum"

`initial.weight.fun` Function for computing initial weights, default NULL means use a random standard normal vector.

Value

Linear model represented as a list of class `aum_linear_model` with named elements: loss is a data table of values for subtrain and optionally validation at each step, weight.vec is the final vector of weights learned via gradient descent, intercept is the value which results in minimal total error (FP+FN), learned via a linear scan over all possible values given the final weight vector, and search is a data table with one row for each step (best step size and number of iterations of line search).

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

`aum_line_search` *aum line search*

Description

Exact line search using a C++ STL map (red-black tree) to implement a queue of line intersection events. If number of rows of `error.diff.df` is B, and number of iterations is I, then space complexity is O(B) and time complexity is O((I+B)log B).

Usage

```
aum_line_search(error.diff.df,
    feature.mat, weight.vec,
    pred.vec = NULL,
    maxIterations = nrow(error.diff.df),
    feature.mat.search = feature.mat,
    error.diff.search = error.diff.df,
    maxStepSize = -1)
```

Arguments

`error.diff.df`	aum_diffs data frame with B rows, one for each breakpoint in example-specific error functions.
`feature.mat`	N x p matrix of numeric features.
`weight.vec`	p-vector of numeric linear model coefficients.
`pred.vec`	N-vector of numeric predicted values. If NULL, `feature.mat` and `weight.vec` will be used to compute predicted values.
`maxIterations`	max number of line search iterations, either a positive integer or "max.auc" or "min.aum" indicating to keep going until AUC decreases or AUM increases.
`feature.mat.search`	feature matrix to use in line search, default is subtrain, can be validation
`error.diff.search`	aum_diffs data frame to use in line search, default is subtrain, can be validation
`maxStepSize`	max step size to explore.

Value

List of class aum_line_search. Element named "line_search_result" is a data table with number of rows equal to `maxIterations` (if it is positive integer, info for all steps, q.size column is number of items in queue at each iteration), otherwise 1 (info for the best step, q.size column is the total number of items popped off the queue).

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

Examples

```
if(require("data.table"))setDTthreads(1L)#for CRAN check.

## Example 1: two binary data.
(bin.diffs <- aum::aum_diffs_binary(c(0,1)))
if(requireNamespace("ggplot2"))plot(bin.diffs)
bin.line.search <- aum::aum_line_search(bin.diffs, pred.vec=c(10,-10))
if(requireNamespace("ggplot2"))plot(bin.line.search)

if(requireNamespace("penaltyLearning")){

  ## Example 2: two changepoint examples, one with three breakpoints.
  data(neuroblastomaProcessed, package="penaltyLearning", envir=environment())
  nb.err <- with(neuroblastomaProcessed$errors, data.frame(
    example=paste0(profile.id, ".", chromosome),
    min.lambda,
    max.lambda,
    fp, fn))
  (nb.diffs <- aum::aum_diffs_penalty(nb.err, c("1.1", "4.2")))
  if(requireNamespace("ggplot2"))plot(nb.diffs)
```

```
nb.line.search <- aum::aum_line_search(nb.diffs, pred.vec=c(1,-1))
if(requireNamespace("ggplot2"))plot(nb.line.search)
aum::aum_line_search(nb.diffs, pred.vec=c(1,-1)-c(1,-1)*0.5)

## Example 3: all changepoint examples, with linear model.
X.sc <- scale(neuroblastomaProcessed$feature.mat)
keep <- apply(is.finite(X.sc), 2, all)
X.subtrain <- X.sc[1:50,keep]
weight.vec <- rep(0, ncol(X.subtrain))
(diffs.subtrain <- aum::aum_diffs_penalty(nb.err, rownames(X.subtrain)))
nb.weight.search <- aum::aum_line_search(
  diffs.subtrain,
  feature.mat=X.subtrain,
  weight.vec=weight.vec,
  maxIterations = 200)
if(requireNamespace("ggplot2"))plot(nb.weight.search)

## Stop line search after finding a (local) max AUC or min AUM.
max.auc.search <- aum::aum_line_search(
  diffs.subtrain,
  feature.mat=X.subtrain,
  weight.vec=weight.vec,
  maxIterations="max.auc")
min.aum.search <- aum::aum_line_search(
  diffs.subtrain,
  feature.mat=X.subtrain,
  weight.vec=weight.vec,
  maxIterations="min.aum")
if(require("ggplot2")){
  plot(nb.weight.search)+
    geom_point(aes(
      step.size, auc),
      data=data.table(max.auc.search[["line_search_result"]], panel="auc"),
      color="red")+
    geom_point(aes(
      step.size, aum),
      data=data.table(min.aum.search[["line_search_result"]], panel="aum"),
      color="red")
}

## Alternate viz with x=iteration instead of step size.
nb.weight.full <- aum::aum_line_search(
  diffs.subtrain,
  feature.mat=X.subtrain,
  weight.vec=weight.vec,
  maxIterations = 1000)
library(data.table)
weight.result.tall <- suppressWarnings(melt(
  nb.weight.full$line_search_result[, iteration:=1:.N][, .(
    iteration, auc, q.size,
    log10.step.size=log10(step.size),
    log10.aum=log10(aum))],
  id.vars="iteration"))
```

```
if(require(ggplot2)){
  ggplot()+
    geom_point(aes(
      iteration, value),
      shape=1,
      data=weight.result.tall)+
    facet_grid(variable ~ ., scales="free")+
    scale_y_continuous("")
}

## Example 4: line search on validation set.
X.validation <- X.sc[101:300,keep]
diffs.validation <- aum::aum_diffs_penalty(nb.err, rownames(X.validation))
valid.search <- aum::aum_line_search(
  diffs.subtrain,
  feature.mat=X.subtrain,
  weight.vec=weight.vec,
  maxIterations = 2000,
  feature.mat.search=X.validation,
  error.diff.search=diffs.validation)
if(requireNamespace("ggplot2"))plot(valid.search)

## validation set max auc, min aum.
max.auc.valid <- aum::aum_line_search(
  diffs.subtrain,
  feature.mat=X.subtrain,
  weight.vec=weight.vec,
  maxIterations="max.auc",
  feature.mat.search=X.validation,
  error.diff.search=diffs.validation)
min.aum.valid <- aum::aum_line_search(
  diffs.subtrain,
  feature.mat=X.subtrain,
  weight.vec=weight.vec,
  maxIterations="min.aum",
  feature.mat.search=X.validation,
  error.diff.search=diffs.validation)
if(require("ggplot2")){
  plot(valid.search)+
    geom_point(aes(
      step.size, auc),
      data=data.table(max.auc.valid[["line_search_result"]], panel="auc"),
      color="red")+
    geom_point(aes(
      step.size, aum),
      data=data.table(min.aum.valid[["line_search_result"]], panel="aum"),
      color="red")
}

## compare subtrain and validation
both.results <- rbind(
  data.table(valid.search$line_search_result, set="validation"),
  data.table(nb.weight.search$line_search_result, set="subtrain"))
```

```
  both.max <- rbind(
    data.table(max.auc.valid$line_search_result, set="validation"),
    data.table(max.auc.search$line_search_result, set="subtrain"))
  ggplot()+
    geom_vline(aes(
      xintercept=step.size, color=set),
      data=both.max)+
    geom_point(aes(
      step.size, auc, color=set),
      shape=1,
      data=both.results)

}
```

`aum_line_search_grid` *aum line search grid*

Description

Line search for predicted values, with grid search to check.

Usage

```
aum_line_search_grid(error.diff.df,
    feature.mat, weight.vec,
    pred.vec = NULL,
    maxIterations = nrow(error.diff.df),
    n.grid = 10L, add.breakpoints = FALSE)
```

Arguments

`error.diff.df`	`aum_diffs` data frame with B rows, one for each breakpoint in example-specific error functions.
`feature.mat`	N x p matrix of numeric features.
`weight.vec`	p-vector of numeric linear model coefficients.
`pred.vec`	N-vector of numeric predicted values. If missing, `feature.mat` and `weight.vec` will be used to compute predicted values.
`maxIterations`	positive int: max number of line search iterations.
`n.grid`	positive int: number of grid points for checking.
`add.breakpoints`	add breakpoints from exact search to grid search.

Value

List of class aum_line_search_grid.

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

Examples

```
if(require("data.table"))setDTthreads(1L)#for CRAN check.

## Example 1: two binary data.
(bin.diffs <- aum::aum_diffs_binary(c(1,0)))
if(requireNamespace("ggplot2"))plot(bin.diffs)
bin.line.search <- aum::aum_line_search_grid(bin.diffs, pred.vec=c(-10,10))
if(requireNamespace("ggplot2"))plot(bin.line.search)

if(requireNamespace("penaltyLearning")){

  ## Example 2: two changepoint examples, one with three breakpoints.
  data(neuroblastomaProcessed, package="penaltyLearning", envir=environment())
  nb.err <- with(neuroblastomaProcessed$errors, data.frame(
    example=paste0(profile.id, ".", chromosome),
    min.lambda,
    max.lambda,
    fp, fn))
  (diffs.subtrain <- aum::aum_diffs_penalty(nb.err, c("4.2", "1.1")))
  if(requireNamespace("ggplot2"))plot(diffs.subtrain)
  (nb.line.search <- aum::aum_line_search_grid(diffs.subtrain, pred.vec=c(-1,1)))
  if(requireNamespace("ggplot2"))plot(nb.line.search)

  ## Example 3: 50 changepoint examples, with linear model.
  X.sc <- scale(neuroblastomaProcessed$feature.mat[1:50,])
  keep <- apply(is.finite(X.sc), 2, all)
  X.subtrain <- X.sc[,keep]
  weight.vec <- rep(0, ncol(X.subtrain))
  diffs.subtrain <- aum::aum_diffs_penalty(nb.err, rownames(X.subtrain))
  nb.weight.search <- aum::aum_line_search_grid(
    diffs.subtrain,
    feature.mat=X.subtrain,
    weight.vec=weight.vec,
    maxIterations = 200)
  if(requireNamespace("ggplot2"))plot(nb.weight.search)

}

## Example 4: counting intersections and intervals at each
## iteration/step size, when there are ties.
(bin.diffs <- aum::aum_diffs_binary(c(0,0,0,1,1,1)))
bin.line.search <- aum::aum_line_search_grid(
  bin.diffs, pred.vec=c(2,3,-1,1,-2,0), n.grid=21)
if(require("ggplot2")){
  plot(bin.line.search)+
    geom_text(aes(
      step.size, Inf, label=sprintf(
```

```
        "%d,%d", intersections, intervals)),
      vjust=1.1,
      data=data.frame(
        panel="threshold", bin.line.search$line_search_result))
}
```

`fn.not.zero` *Penalized models with non-zero fn at penalty=0*

Description

Usually we assume that fn must be zero at penalty=0, but this is not always the case in real data/labels. For example in the PeakSegDisk model with penalty=0, there are peaks almost everywhere but if a positive label is too small or misplaced with respect to the detected peaks, then there can be false negatives.

Usage

```
data("fn.not.zero")
```

Format

A data frame with 156 observations on the following 5 variables.

`example` a character vector

`min.lambda` a numeric vector

`max.lambda` a numeric vector

`fp` a numeric vector

`fn` a numeric vector

Source

https://github.com/tdhock/feature-learning-benchmark

`neg.zero.fp` *Negative zero FP*

Description

A data set that resulted in an error, negative FP, but actually numerically zero.

Usage

```
data("neg.zero.fp")
```

Format

Named list. diffs is a data table, output of aum_diffs, pred is a numeric vector of predictions.

plot.aum_diffs *plot aum diffs*

Description

Plot method for aum_diffs which shows piecewise constant error functions. Uses aum_errors internally to compute error functions which are plotted. Not recommended for large number of examples (>20).

Usage

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

Arguments

x	data table with class "aum_diffs".
...	ignored.

Value

ggplot of error functions, each example in a different panel.

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

plot.aum_line_search *plot aum line search*

Description

Plot method for aum_line_search which shows AUM and threshold functions.

Usage

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

Arguments

x	list with class "aum_line_search".
...	ignored.

Value

ggplot.

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

`plot.aum_line_search_grid` *plot aum line search grid*

Description

Plot method for `aum_line_search_grid` which shows AUM and threshold functions, along with grid points for checking.

Usage

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

Arguments

x	list with class "aum_line_search_grid".
...	ignored.

Value

ggplot.

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

`set_loss_plot` *set loss plot*

Description

plot subtrain/validation loss.

Usage

```
set_loss_plot(loss.dt,
    set.colors = c(subtrain = "black",
        validation = "red"))
```

Arguments

`loss.dt`	loss.dt
`set.colors`	set.colors

Author(s)

Toby Dylan Hocking <toby.hocking@r-project.org> [aut, cre], Jadon Fowler [aut] (Contributed exact line search C++ code)

Index