Package 'FeatureHashing'

January 1, 2015

Type Package

Title Creates a Model Matrix via Feature Hashing With a Formula Interface

Version 0.8

Date 2014-08-01

Author Wush Wu [aut, cre]

Maintainer Wush Wu <wush978@gmail.com>

Description Feature hashing, also called as the hashing trick, is a method to transform features to vector. Without looking up the indices in an associative array, it applies a hash function to the features and uses their hash values as indices directly. The method of feature hashing in this package was proposed in Weinberger et. al. (2009). The hashing algorithm is the murmurhash3 from the digest package. Please see the README.md for more information.

License GPL (>= 3)

Depends R (>= 3.1), methods

Imports Rcpp (>= 0.11), Matrix, digest(>= 0.6.8)

LinkingTo Rcpp, digest(>= 0.6.8), BH

Suggests pack, RUnit

SystemRequirements C++11

BugReports https://github.com/wush978/FeatureHashing/issues

URL https://github.com/wush978/FeatureHashing

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-01-01 06:05:43

R topics documented:

`CSCMatrix-class` *CSCMatrix*

Description

The structure of `CSCMatrix` is the same as the structure of `dgCMatrix`. However, the `CSCMatrix` has weaker constraints compared to `dgCMatrix`.

`CSCMatrix` onlysupports limited operators. The users can convert it to `dgCMatrix` for compatibility of existed algorithms.

Details

The `CSCMatrix` violates two constraints used in `dgCMatrix`:

- The row indices should be sorted with columns.
- The row indices should be unique with columns.

The result of matrix-vector multiplication should be the same.

Methods

- `dim` The dimension of the matrix object `CSCMatrix`.
- `dim<-` The assignment of dimension of the matrix object `CSCMatrix`.
- `[` The subsetting operator of the matrix object `CSCMatrix`.
- `%*%` The matrix-vector multiplication of the matrix object `CSCMatrix`. The returned object is a numeric vector.

See Also

`dgCMatrix-class`

Examples

```
# construct a CSCMatrix
m <- hashed.model.matrix(~ ., CO2, 8)
# convert it to dgCMatrix
m2 <- as(m, "dgCMatrix")
```

hashed.model.matrix *Create a model matrix with feature hashing*

Description

Create a model matrix with feature hashing

Usage

```
hashed.model.matrix(object, data, hash_size = 2^24, transpose = TRUE,
  keep.hashing_mapping = FALSE)
```

Arguments

object	formula. A model formula.
data	data.frame. The original data.
hash_size	positive integer. The hash size of feature hashing.
transpose	logical value. Indicating if the transpose should be returned.
keep.hashing_mapping	logical value. The indicator of whether storing the hash mapping or not.

Details

The hashed.model.matrix hashes the feature automatically during the construction of the model matrix. It uses the 32-bit variant of MurmurHash3 https://code.google.com/p/smhasher/wiki/MurmurHash3. Weinberger et. al. (2009) used two separate hashing function h(hash_h) and ξ(hash_xi) to determine the indices and the sign of the values respectively. Different seeds are used to implement the hashing function h and ξ with MurmurHash3.

The object formula is parsed via terms.formula with "tag" as special keyword. The interaction term is hashed in different ways. Please see example for the detailed implementation. The "tag" is used to expand the concatenated feature such as "1,27,19,25,tp,tw" which represents the occurrence of multiple categorical variable. The hashed.model.matrix will expand the tag feature and produce the related model matrix.

The "tag" accepts two parameters:

- split, character value used for splitting.
- type, one of existence or count.

The user could explore the behavior via function tag.

References

Kilian Q. Weinberger, Anirban Dasgupta, John Langford, Alexander J. Smola, and Josh Attenberg. ICML, volume 382 of ACM International Conference Proceeding Series, page 140. ACM, (2009)

Examples

```
# Construct the model matrix. The transposed matrix is returned by default.
m <- hashed.model.matrix(~ ., CO2, 2^6, keep.hashing_mapping = TRUE)
# Print the matrix via dgCMatrix
as(m, "dgCMatrix")
# Check the result of hashing
mapping <- unlist(as.list(attr(m, "mapping")))
# Check the rate of collision
# mean(duplicated(mapping %% 2^6))
# The result is CSCMatrix which supports simple subsetting and matrix-vector
# multiplication
# rnorm(2^6) %*% m

# Detail of the hashing
## The main effect is hashed via `hash_h`
all(hash_h(names(mapping)) %% 2^6 == mapping %% 2^6)
## The sign is corrected by `hash_xi`
hash_xi(names(mapping))
## The interaction term is implemented as follow:
m2 <- hashed.model.matrix(~ .^2, CO2, 2^6, keep.hashing_mapping = TRUE)
mapping2 <- unlist(as.list(attr(m2, "mapping")))
mapping2[2] # PlantQn2:uptake
h1 <- mapping2["PlantQn2"]
h2 <- mapping2["uptake"]
library(pack)
hash_h(rawToChar(c(numToRaw(h1, 4), numToRaw(h2, 4)))) # should be mapping2[2]

# The tag-like feature
data(test.tag)
df <- data.frame(a = test.tag, b = rnorm(length(test.tag)))
m <- hashed.model.matrix(~ tag(a, split = ",", type = "existence"):b, df, 2^6,
 keep.hashing_mapping = TRUE)
# The column `a` is splitted by "," and have an interaction with "b":
mapping <- unlist(as.list(attr(m, "mapping")))
names(mapping)
```

---

tag *Expand concatenated feature*

Description

Expand concatenated feature

Usage

```
tag(x, split = ",", type = c("count", "existence"))
```

Arguments

x	character vector or factor. The source of tag features.
split	character vector. The split symbol for tag features.
type	character value. Either "count" or "existence". "count" indicates the number of occurrence of the tag. "existence" indicates the boolean that whether the tag exist or not.

Value

integer vector for `type = "count"` and logical vector for `type = "existence"`.

`test.tag` *test.tag*

Description

This is a vector to demo the concatenated feature.

Usage

```
test.tag
```

Format

For each element, the string represents the occurrence of different tags. For example, the string "1,27,19,25,tp,tw" of the first instance represents that the feature ‘1‘ is TRUE, the feature ‘27‘ is TRUE, et. al. On the contrary, the missing feature such as ‘2‘ is FALSE.

Index