Package 'FNN'

January 20, 2025

Version 1.1.4.1

Date 2023-12-31

Title Fast Nearest Neighbor Search Algorithms and Applications

Depends R (>= 4.0.0)

Suggests chemometrics, mvtnorm

Description Cover-tree and kd-tree fast k-nearest neighbor search algorithms and related applications including KNN classification, regression and information measures are implemented.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-09-22 08:24:48 UTC

Author Alina Beygelzimer [aut] (cover tree library),
Sham Kakadet [aut] (cover tree library),
John Langford [aut] (cover tree library),
Sunil Arya [aut] (ANN library 1.1.2 for the kd-tree approach),
David Mount [aut] (ANN library 1.1.2 for the kd-tree approach),
Shengqiao Li [aut, cre]

Maintainer Shengqiao Li <lishengqiao@yahoo.com>

Contents

`crossentropy` *Cross Entropy*

Description

KNN Cross Entropy Estimators.

Usage

```
crossentropy(X, Y, k=10, algorithm=c("kd_tree", "cover_tree", "brute"))
```

Arguments

`X`	an input data matrix.
`Y`	an input data matrix.
`k`	the maximum number of nearest neighbors to search. The default value is set to 10.
`algorithm`	nearest neighbor search algorithm.

Details

If `p(x)` and `q(x)` are two continuous probability density functions, then the cross-entropy of `p` and `q` is defined as $H(p;q) = E_p[-\log q(x)]$.

Value

a vector of length `k` for crossentropy estimates using `1:k` nearest neighbors, respectively.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: `<lishengqiao@yahoo.com>`

References

S. Boltz, E. Debreuve and M. Barlaud (2007). "kNN-based high-dimensional Kullback-Leibler distance for tracking". *Image Analysis for Multimedia Interactive Services, 2007. WIAMIS '07. Eighth International Workshop on*.

`entropy` *Shannon Entropy*

Description

KNN Shannon Entropy Estimators.

Usage

```
entropy(X, k = 10, algorithm = c("kd_tree", "brute"))
```

Arguments

`X`	an input data matrix.
`k`	the maximum number of nearest neighbors to search. The default value is set to 10.
`algorithm`	nearest neighbor search algorithm.

Value

a vector of length `k` for entropy estimates using `1:k` nearest neighbors, respectively.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: <lishengqiao@yahoo.com>

References

H. Singh, N. Misra, V. Hnizdo, A. Fedorowicz and E. Demchuk (2003). "Nearest neighbor estimates of entropy". *American Journal of Mathematical and Management Sciences*, **23**, 301-321.

M.N. Goria, N.N.Leonenko, V.V. Mergel and P.L. Novi Inverardi (2005). "A new class of random vector entropy estimators and its applications in testing statistical hypotheses". *Journal of Nonparametric Statistics*, **17**:3, 277–297.

R.M. Mnatsakanov, N. Misra, S. Li and E.J. Harner (2008). "K_n-nearest neighbor estimators of entropy". *Mathematical Methods of Statistics*, **17**:3, 261-277.

`get.knn` *Search Nearest Neighbors*

Description

Fast k-nearest neighbor searching algorithms including a kd-tree, cover-tree and the algorithm implemented in class package.

Usage

```
get.knn(data, k=10, algorithm=c("kd_tree", "cover_tree", "CR", "brute"))
get.knnx(data, query, k=10, algorithm=c("kd_tree", "cover_tree",
"CR", "brute"))
```

Arguments

`data`	an input data matrix.
`query`	a query data matrix.
`algorithm`	nearest neighbor searching algorithm.
`k`	the maximum number of nearest neighbors to search. The default value is set to 10.

Details

The *cover tree* is O(n) space data structure which allows us to answer queries in the same O(log(n)) time as *kd tree* given a fixed intrinsic dimensionality. Templated code from https://hunch.net/~jl/projects/cover_tree/cover_tree.html is used.

The *kd tree* algorithm is implemented in the Approximate Near Neighbor (ANN) C++ library (see http://www.cs.umd.edu/~mount/ANN/). The exact nearest neighbors are searched in this package.

The *CR* algorithm is the *VR* using distance *1-x'y* assuming `x` and `y` are unit vectors. The *brute* algorithm searches linearly. It is a naive method.

Value

a list contains:

`nn.index`	an n x k matrix for the nearest neighbor indice.
`nn.dist`	an n x k matrix for the nearest neighbor Euclidean distances.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: <lishengqiao@yahoo.com>

References

Bentley J.L. (1975), "Multidimensional binary search trees used for associative search," *Communication ACM*, **18**, 309-517.

Arya S. and Mount D.M. (1993), "Approximate nearest neighbor searching," *Proc. 4th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*, 271-280.

Arya S., Mount D.M., Netanyahu N.S., Silverman R. and Wu A.Y. (1998), "An optimal algorithm for approximate nearest neighbor searching," *Journal of the ACM*, **45**, 891-923.

Beygelzimer A., Kakade S. and Langford J. (2006), "Cover trees for nearest neighbor," *ACM Proc. 23rd international conference on Machine learning*, **148**, 97-104.

See Also

nn2 in **RANN**, ann in **yaImpute** and knn in **class**.

Examples

```
data<- query<- cbind(1:10, 1:10)

get.knn(data, k=5)
get.knnx(data, query, k=5)
get.knnx(data, query, k=5, algo="kd_tree")

th<- runif(10, min=0, max=2*pi)
data2<-  cbind(cos(th), sin(th))
get.knn(data2, k=5, algo="CR")
```

KL.dist *Kullback-Leibler Divergence*

Description

Compute Kullback-Leibler symmetric distance.

Usage

```
KL.dist(X, Y, k = 10, algorithm=c("kd_tree", "cover_tree", "brute"))
KLx.dist(X, Y, k = 10, algorithm="kd_tree")
```

Arguments

X	An input data matrix.
Y	An input data matrix.
k	The maximum number of nearest neighbors to search. The default value is set to 10.
algorithm	nearest neighbor search algorithm.

Details

Kullback-Leibler distance is the sum of divergence `q(x)` from `p(x)` and `p(x)` from `q(x)` .

`KL.*` versions return distances from `C` code to `R` but `KLx.*` do not.

Value

Return the Kullback-Leibler distance between `X` and `Y`.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: `<lishengqiao@yahoo.com>`

References

S. Boltz, E. Debreuve and M. Barlaud (2007). "kNN-based high-dimensional Kullback-Leibler distance for tracking". *Image Analysis for Multimedia Interactive Services, 2007. WIAMIS '07. Eighth International Workshop on*.

S. Boltz, E. Debreuve and M. Barlaud (2009). "High-dimensional statistical measure for region-of-interest tracking". *Trans. Img. Proc.*, **18**:6, 1266–1283.

See Also

`KL.divergence`.

Examples

```
set.seed(1000)
X<- rexp(10000, rate=0.2)
Y<- rexp(10000, rate=0.4)

KL.dist(X, Y, k=5)
KLx.dist(X, Y, k=5)
#thoretical distance = (0.2-0.4)^2/(0.2*0.4) = 0.5
```

KL.divergence *Kullback-Leibler Divergence*

Description

Compute Kullback-Leibler divergence.

Usage

```
KL.divergence(X, Y, k = 10, algorithm=c("kd_tree", "cover_tree", "brute"))
KLx.divergence(X, Y, k = 10, algorithm="kd_tree")
```

Arguments

X	An input data matrix.
Y	An input data matrix.
k	The maximum number of nearest neighbors to search. The default value is set to 10.
algorithm	nearest neighbor search algorithm.

Details

If p(x) and q(x) are two continuous probability density functions, then the Kullback-Leibler divergence of q from p is defined as $E_p[\log \frac{p(x)}{q(x)}]$.

KL.* versions return divergences from C code to R but KLx.* do not.

Value

Return the Kullback-Leibler divergence from X to Y.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: <lishengqiao@yahoo.com>

References

S. Boltz, E. Debreuve and M. Barlaud (2007). "kNN-based high-dimensional Kullback-Leibler distance for tracking". *Image Analysis for Multimedia Interactive Services, 2007. WIAMIS '07. Eighth International Workshop on*.

S. Boltz, E. Debreuve and M. Barlaud (2009). "High-dimensional statistical measure for region-of-interest tracking". *Trans. Img. Proc.*, **18**:6, 1266–1283.

See Also

KL.dist

Examples

```
set.seed(1000)
X<- rexp(10000, rate=0.2)
Y<- rexp(10000, rate=0.4)

KL.divergence(X, Y, k=5)
#theoretical divergence = log(0.2/0.4)+(0.4/0.2)-1 = 1-log(2) = 0.307
```

knn *k-Nearest Neighbour Classification*

Description

k-nearest neighbour classification for test set from training set. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.

Usage

```
knn(train, test, cl, k = 1, prob = FALSE, algorithm=c("kd_tree",
    "cover_tree", "brute"))
```

Arguments

train	matrix or data frame of training set cases.
test	matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.
cl	factor of true classifications of training set.
k	number of neighbours considered.
prob	if this is true, the proportion of the votes for the winning class are returned as attribute prob.
algorithm	nearest neighbor search algorithm.

Value

factor of classifications of test set. doubt will be returned as NA.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: <lishengqiao@yahoo.com>

References

B.D. Ripley (1996). *Pattern Recognition and Neural Networks.* Cambridge.

M.N. Venables and B.D. Ripley (2002). *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

ownn, knn.cv and knn in **class**.

Examples

```
data(iris3)
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
knn(train, test, cl, k = 3, prob=TRUE)
attributes(.Last.value)
```

---

knn.cv *k-Nearest Neighbour Classification Cross-Validation*

Description

k-nearest neighbour classification cross-validation from training set.

Usage

```
knn.cv(train, cl, k = 1, prob = FALSE, algorithm=c("kd_tree",
       "cover_tree", "brute"))
```

Arguments

train	matrix or data frame of training set cases.
cl	factor of true classifications of training set
k	number of neighbours considered.
prob	if this is true, the proportion of the votes for the winning class are returned as attribute prob.
algorithm	nearest neighbor search algorithm.

Details

This uses leave-one-out cross validation. For each row of the training set train, the k nearest (in Euclidean distance) other training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote.

Value

factor of classifications of training set. doubt will be returned as NA. distances and indice of k nearest neighbors are also returned as attributes.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: <lishengqiao@yahoo.com>

References

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks.* Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* Fourth edition. Springer.

See Also

knn and knn.cv in **class**.

Examples

```
data(iris3)
train <- rbind(iris3[,,1], iris3[,,2], iris3[,,3])
cl <- factor(c(rep("s",50), rep("c",50), rep("v",50)))
knn.cv(train, cl, k = 3, prob = TRUE)
attributes(.Last.value)
```

---

knn.dist *k Nearest Neighbor Distances*

Description

Fast k-nearest neighbor distance searching algorithms.

Usage

```
knn.dist(data, k=10, algorithm=c("kd_tree", "cover_tree", "CR", "brute"))
knnx.dist(data, query, k=10, algorithm=c("kd_tree", "cover_tree",
          "CR", "brute"))
```

Arguments

data	an input data matrix.
query	a query data matrix.
algorithm	nearest neighbor searching algorithm.
k	the maximum number of nearest neighbors to search. The default value is set to 10.

Value

return the Euclidiean distances of k nearest neighbors.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: <lishengqiao@yahoo.com>

References

Bentley J.L. (1975), "Multidimensional binary search trees used for associative search," *Communication ACM*, **18**, 309-517.

Arya S. and Mount D.M. (1993), "Approximate nearest neighbor searching," *Proc. 4th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*, 271-280.

Arya S., Mount D.M., Netanyahu N.S., Silverman R. and Wu A.Y. (1998), "An optimal algorithm for approximate nearest neighbor searching," *Journal of the ACM*, **45**, 891-923.

Beygelzimer A., Kakade S. and Langford J. (2006), "Cover trees for nearest neighbor," *ACM Proc. 23rd international conference on Machine learning*, **148**, 97-104.

See Also

`get.knn` and `knn.index` .

Examples

```
if(require(mvtnorm))
{
  sigma<- function(v, r, p)
  {
     V<- matrix(r^2, ncol=p, nrow=p)
     diag(V)<- 1
      V*v
  }

  X<- rmvnorm(1000, mean=rep(0, 20), sigma(1, .5, 20))
  print(system.time(knn.dist(X)) )
  print(system.time(knn.dist(X, algorithm = "kd_tree")))

}
```

`knn.index` *Search Nearest Neighbors*

Description

Fast k-nearest neighbor searching algorithms including a kd-tree, cover-tree and the algorithm implemented in class package.

Usage

```
knn.index(data, k=10, algorithm=c("kd_tree", "cover_tree", "CR", "brute"))
knnx.index(data, query, k=10, algorithm=c("kd_tree", "cover_tree",
           "CR", "brute"))
```

Arguments

data	an input data matrix.
query	a query data matrix.
algorithm	nearest neighbor searching algorithm.
k	the maximum number of nearest neighbors to search. The default value is set to 10.

Value

return the indice of k nearest neighbors.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: <lishengqiao@yahoo.com>

References

Bentley J.L. (1975), "Multidimensional binary search trees used for associative search," *Communication ACM*, **18**, 309-517.

Arya S. and Mount D.M. (1993), "Approximate nearest neighbor searching," *Proc. 4th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*, 271-280.

Arya S., Mount D.M., Netanyahu N.S., Silverman R. and Wu A.Y. (1998), "An optimal algorithm for approximate nearest neighbor searching," *Journal of the ACM*, **45**, 891-923.

Beygelzimer A., Kakade S. and Langford J. (2006), "Cover trees for nearest neighbor," *ACM Proc. 23rd international conference on Machine learning*, **148**, 97-104.

See Also

knn.dist and get.knn.

Examples

```
data<- query<- cbind(1:10, 1:10)

knn.index(data, k=5)
knnx.index(data, query, k=5)
knnx.index(data, query, k=5, algo="kd_tree")
```

knn.reg *k Nearest Neighbor Regression*

Description

k-nearest neighbor regression

Usage

```
knn.reg(train, test = NULL, y, k = 3, algorithm=c("kd_tree",
        "cover_tree", "brute"))
```

Arguments

train	matrix or data frame of training set cases.
test	matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case. If not supplied, cross-validataion will be done.
y	reponse of each observation in the training set.
k	number of neighbours considered.
algorithm	nearest neighbor search algorithm.

Details

If test is not supplied, Leave one out cross-validation is performed and *R-square* is the predicted R-square.

Value

knn.reg returns an object of class "knnReg" or "knnRegCV" if test data is not supplied.

The returnedobject is a list containing at least the following components:

call	the match call.
k	number of neighbours considered.
n	number of predicted values, either equals test size or train size.
pred	a vector of predicted values.
residuals	predicted residuals. NULL if test is supplied.
PRESS	the sums of squares of the predicted residuals. NULL if test is supplied.
R2Pred	predicted R-square. NULL if test is supplied.

Note

The code for "VR" nearest neighbor searching is taken from class source

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: <lishengqiao@yahoo.com>

See Also

knn.

Examples

```
if(require(chemometrics)){
  data(PAC);
  pac.knn<- knn.reg(PAC$X, y=PAC$y, k=3);

  plot(PAC$y, pac.knn$pred, xlab="y", ylab=expression(hat(y)))
}
```

`mutinfo` *Mutual Information*

Description

KNN Mutual Information Estimators.

Usage

```
mutinfo(X, Y, k=10, direct=TRUE)
```

Arguments

`X`	an input data matrix.
`Y`	an input data matrix.
`k`	the maximum number of nearest neighbors to search. The default value is set to 10.
`direct`	Directly compute or via entropies.

Details

The direct computation is based on the first estimator of A. Kraskov, H. Stogbauer and P.Grassberger (2004) and the indirect computation is done via entropy estimates, i.e., I(X, Y) = H (X) + H(Y) - H(X, Y). The direct method has smaller bias and variance but the indirect method is faster, see Evans (2008).

Value

For direct method, one mutual information estimate; For indirect method,a vector of length `k` for mutual information estimates using `1:k` nearest neighbors, respectively.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: <lishengqiao@yahoo.com>

References

A. Kraskov, H. Stogbauer and P.Grassberger (2004). "Estimating mutual information". *Physical Review E*, **69**:066138, 1–16.

D. Evans (2008). "A Computationally efficient estimator for mutual information". *Proc. R. Soc. A*, **464**, 1203–1215.

---

ownn *Optimal Weighted Nearest Neighbor Classification*

Description

This function implements Samworth's optimal weighting scheme for k nearest neighbor classification. The performance improvement is greatest when the dimension is 4 as reported in the reference.

Usage

```
ownn(train, test, cl, testcl=NULL, k = NULL, prob = FALSE,
    algorithm=c("kd_tree", "cover_tree", "brute"))
```

Arguments

train	matrix or data frame of training set cases.
test	matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.
cl	factor of true classifications of training set.
testcl	factor of true classifications of testing set for error rate calculation.
k	number of neighbours considered, chosen by 5-fold cross-validation if not supplied.
prob	if this is true, the proportion of the weights for the winning class are returned as attribute prob.
algorithm	nearest neighbor search algorithm.

Value

a list includes k, predictions by ordinary knn, optimal weighted knn and bagged knn, and accuracies if class labels of test data set are given.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: <lishengqiao@yahoo.com>

References

Richard J. Samworth (2012), "Optimal Weighted Nearest Neighbor Classifiers," *Annals of Statistics*, **40:5**, 2733-2763.

See Also

knn and knn in **class**.

Examples

```
data(iris3)
train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
testcl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
out <- ownn(train, test, cl, testcl)
out
```

---

print.knnReg *Print Method for KNN Regression*

Description

Print method for KNN regression.

Usage

```
## S3 method for class 'knnReg'
print(x, ...)
## S3 method for class 'knnRegCV'
print(x, ...)
```

Arguments

x	a knnReg or knnRegCV object.
...	Additonal print arguments.

Author(s)

Shengqiao Li. To report any bugs or suggestions please email: <lishengqiao@yahoo.com>

Index