Package 'GACFF'

January 20, 2025

Type Package

Title Genetic Similarity in User-Based Collaborative Filtering

Version 1.0

Date 2019-12-08

Depends R (>= 3.0.0), graphics, stats, utils

Encoding UTF-8

Author Farimah Houshmand Nanehkaran, Seyed Mohammad Reza Lajevardi <R.Lajevardi@iaukashan.ac.ir>, Mahmoud Mahlouji Bidgholi <m.mahlouji@iaukashan.ac.ir>

Maintainer Farimah Houshmand Nanehkaran <hoshmandcomputer@gmail.com>

Description The genetic algorithm can be used directly to find the similarity of users and more effectively to increase the efficiency of the collaborative filtering method.
By identifying the nearest neighbors to the active user, before the genetic algorithm, and by identifying suitable starting points, an effective method for user-based collaborative filtering method has been developed.
This package uses an optimization algorithm (continuous genetic algorithm) to directly find the optimal similarities between active users (users for whom current recommendations are made) and others.
First, by determining the nearest neighbor and their number, the number of genes in a chromosome is determined. Each gene represents the neighbor's similarity to the active user.
By estimating the starting points of the genetic algorithm, it quickly converges to the optimal solutions.
The positive point is the independence of the genetic algorithm on the number of data that for big data is an effective help in solving the problem.

License GPL (>= 2)

RoxygenNote 7.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-12-20 13:50:11 UTC

Contents

`GACFF-package` *Genetic Similarity in User-Based Collaborative Filtering*

Description

The genetic algorithm can be used directly to find the similarity of users and more effectively to increase the efficiency of the collaborative filtering method. By identifying the nearest neighbors to the active user, before the genetic algorithm, and by identifying suitable starting points, an effective method for user-based collaborative filtering method has been developed. This package uses an optimization algorithm (continuous genetic algorithm) to directly find the optimal similarities between active users (users for whom current recommendations are made) and others. First, by determining the nearest neighbor and their number, the number of genes in a chromosome is determined. Each gene represents the neighbor's similarity to the active user. By estimating the starting points of the genetic algorithm, it quickly converges to the optimal solutions. The positive point is the independence of the genetic algorithm on the number of data that for big data is an effective help in solving the problem.

Details

The DESCRIPTION file:

Package:	GACFF
Type:	Package
Title:	Genetic Similarity in User-Based Collaborative Filtering
Version:	1.0
Date:	2019-12-08
Depends:	R (>= 3.0.0), graphics, stats, utils
Encoding:	UTF-8
Author:	Farimah Houshmand Nanehkaran, Seyed Mohammad Reza Lajevardi <R.Lajevardi@iaukashan.ac.ir>, Mahn
Maintainer:	Farimah Houshmand Nanehkaran <hoshmandcomputer@gmail.com>
Description:	The genetic algorithm can be used directly to find the similarity of users and more effectively to increase the
License:	GPL (>= 2)
RoxygenNote:	7.0.1

Index of help topics:

Genetic-based recommender systems.

Finding the Nearest Neighbors and Using Them in the Genetic-Based Collaborative Filtering Recommender System.

Author(s)

Farimah Houshmand Nanehkaran, Seyed Mohammad Reza Lajevardi <R.Lajevardi@iaukashan.ac.ir>, Mahmoud Mahlouji Bidgholi <m.mahlouji@iaukashan.ac.ir>

Maintainer: Farimah Houshmand Nanehkaran <hoshmandcomputer@gmail.com>

References

Bobadilla, J., Ortega, F., Hernando, A. and Alcalá, J. (2011). *Improving collaborative filtering recommender system results and performance using genetic algorithms*. Knowledge-based systems, vol. 24, no. 8, pp. 1310-1316.

Ben-Shimon, D., Rokach, L. and Shapira, B. (2016). *An ensemble method for top-N recommendations from the SVD*. Expert Systems with Applications, vol. 64, pp.84-92.

Kang, Z., Peng, C. and Cheng, Q. (2016). *Top-n recommender system via matrix completion*. In Thirtieth AAAI Conference on Artificial Intelligence.

Qian, Y., Zhang, Y., Ma, X., Yu, H. and Peng, L. (2019). *EARS: Emotion-aware recommender system based on hybrid information fusion*. Information Fusion, vol. 46, pp.141-146.

Xia, B., Li, T., Li, Q. and Zhang, H. (2018). *Noise-tolerance matrix completion for location recommendation*. Data Mining and Knowledge Discovery, vol. 32, no. 1, pp.1-24.

Examples

```
ratings <- matrix(c(  2,    5,  NaN,  NaN,  NaN,    4,
                    NaN,  NaN,  NaN,    1,  NaN,    5,
                    NaN,    4,    5,  NaN,    4,  NaN,
```

```
                      4,  NaN,  NaN,    5,  NaN,  NaN,
                      5,  NaN,    2,  NaN,  NaN,  NaN,
                    NaN,    1,  NaN,    4,    2,  NaN),nrow=6,byrow=TRUE)

active_users <- c(1:dim(ratings)[2])
##1
sim.Pearson  <- Similarity_Pearson (ratings, active_user=6,
                                    near_user=c(1:dim(ratings)[2]))
##2
Pearson.out  <- Pearson (ratings, active_user=6, Threshold_KNN=4)
##3
predict      <-Prediction (ratings, active_user=6,
                         near_user=Pearson.out$near_user_Pearson,
                         sim_x=Pearson.out$sim_Pearson,
                         KNN=length(Pearson.out$sim_Pearson))
##4
ItemSelect (ratings, active_user=6, pre_x=predict)
##5
NewKNN.out  <- NewKNN (ratings, active_user=6, Threshold_KNN=4,
                         max_scour=5, min_scour=1)
##6
Genetic.out <- Genetic (ratings, active_user=6,
                          near_user=NewKNN.out$near_user,
                          Threshold_KNN=4, max_scour=5, min_scour=1,
                          PopSize=100, MaxIteration=50, CrossPercent=70,
                          MutatPercent=20)
##7
Results.out <- Results(ratings, active_users, Threshold_KNN=4, max_scour=5,
                       min_scour=1, PopSize=100, MaxIteration=50,
                       CrossPercent=70, MutatPercent=20)
##8
meanR.Results.out <- meanR.Results (obj_Results=Results.out)
##9
plotResults(active_users, Results.out, xlab = "Iteration", ylab = "MAE",
                  main = "MAE (New KNN+GA) in CF Recommender Systems" )
```

`Genetic` *The genetic algorithm for finding similarities between users.*

Description

Finding users' similarity by continuous genetic algorithm directly.

Usage

```
Genetic(ratings, active_user, near_user, Threshold_KNN, max_scour, min_scour,
         PopSize=100, MaxIteration=50, CrossPercent=70, MutatPercent=20)
```

Arguments

ratings	A rating matrix whose rows are items and columns are users.
active_user	The id of an active user as an integer greater than zero (for example active_user<-6).
near_user	The number of neighbor users that obtained from "NewKNN" for the active user.
Threshold_KNN	Maximum number of neighbors.
max_scour	The maximum range of ratings.
min_scour	The minimum range of ratings.
PopSize	Population size (Number of chromosomes) in Genetic algorithm.
MaxIteration	Number of iterations in Genetic algorithm.
CrossPercent	Percentage of the Genetic algorithm population that participates in the Single-point crossover operator to generate new offspring.
MutatPercent	Percentage of the Genetic algorithm population that participates in the mutation.

Details

The fitness function of the genetic algorithm determines the optimality of the neighbor's similarity to the active user. The fitness function is considered the MAE of the RS. The MAE is obtained by comparing the real ratings of users with the predicted ratings that are calculated according to the similarity obtained by the Genetic algorithm.

The steps of the Genetic algorithm are:

Selection. Selection is based on elitism. Using this operator, the best member of each population survives and will be present in the next population. In other words, the member with the highest match will automatically be transferred to the new population (elitist $selection = 10\%$ of the best individuals from each generation). The application of elitism in the genetic algorithm usually improves its efficiency.

Crossover. Single-point crossover technique is used. The crossover operator is used to produce children. A weight coefficient (crossover probability) of between 0 and 1 (0.8) is considered equal to the length of the parent, and by using follow formulas, two new chromosomes or two children are created.

$$y1 = \alpha * x1 + (1 - \alpha) * x2$$
$$y2 = \alpha * x2 + (1 - \alpha) * x1$$

$x1$ and $x2$ are decimal values that represent the parent chromosome. α is the weighting factor and, $y1$ and $y2$ are the children's chromosomes resulting from the parent compound.

Mutation. Single-point mutation technique is used to introduce diversity. The mutation probability is 0.02. The Gaussian mutation operator is implemented. First, a chromosome is randomly selected from the population, and then one or more of its components is changed according to the Gaussian function using follow formula.

$$y1 = x1 + r1 * N(0, 1)$$

$x1$ is the similarity value that represents the parent chromosome, $r1$ is a random number in the range of 0 and 1 (0.02). N(0,1) is a random number distributed by using the Gaussian distribution.

The genetic algorithm stops when an individual in the population has a fitness value less than a constant value (for example 0.5).

Value

An object of class "Genetic", a list with components:

call	The call used.
sim_GA	Similarity obtained from the Genetic algorithm as much as the number of neighbors.
pre_GA	Predicted active user ratings for all items.
item_GA	A set of best-predicted items for the active user.
save_MAE_GA	A set of MAEs obtained from each iteration of the Genetic algorithm.
time_Genetic	The elapsed time of Genetic algorithm.

Author(s)

Farimah Houshmand Nanehkaran

Maintainer: Farimah Houshmand Nanehkaran <hoshmandcomputer@gmail.com>

References

Bobadilla, J., Ortega, F., Hernando, A. and Alcala, J. (2011). *Improving collaborative filtering recommender system results and performance using genetic algorithms*. Knowledge-based systems, vol. 24, no. 8, pp. 1310-1316.

Examples

```
ratings <- matrix(c(  2,    5,  NaN,  NaN,  NaN,    4,
                    NaN,  NaN,  NaN,    1,  NaN,    5,
                    NaN,    4,    5,  NaN,    4,  NaN,
                      4,  NaN,  NaN,    5,  NaN,  NaN,
                      5,  NaN,    2,  NaN,  NaN,  NaN,
                    NaN,    1,  NaN,    4,    2,  NaN),nrow=6,byrow=TRUE)

NewKNN.out  <- NewKNN (ratings, active_user=6, Threshold_KNN=4,
                       max_scour=5, min_scour=1)

Genetic.out <- Genetic (ratings, active_user=6,
                        near_user=NewKNN.out$near_user,
                        Threshold_KNN=4, max_scour=5, min_scour=1,
                        PopSize=100, MaxIteration=50, CrossPercent=70,
                        MutatPercent=20)
```

ItemSelect *A set of Items id for recommending to an active user.*

Description

Selecting the best items to recommend.

Usage

```
ItemSelect(ratings, active_user, pre_x)
```

Arguments

ratings	A rating matrix whose rows are items and columns are users.
active_user	The id of an active user as an integer greater than zero (for example active_user<-6).
pre_x	A set of predicted ratings for all items not rated by the active user.

Details

Items selecting and their order depends on the method (Pearson, NewKNN, Genetic).

Value

item_x	A set of item identifiers recommended to the active user.

References

Nilashi, M., Ibrahim, O. and Bagherifard, K. (2018). *A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques*. Expert Systems with Applications, vol. 92, pp. 507-520.

Examples

```
ratings <- matrix(c(  2,    5,  NaN,  NaN,  NaN,    4,
                    NaN,  NaN,  NaN,    1,  NaN,    5,
                    NaN,    4,    5,  NaN,    4,  NaN,
                      4,  NaN,  NaN,    5,  NaN,  NaN,
                      5,  NaN,    2,  NaN,  NaN,  NaN,
                    NaN,    1,  NaN,    4,    2,  NaN),nrow=6,byrow=TRUE)

Pearson.out  <- Pearson (ratings, active_user=6, Threshold_KNN=4)

predict      <-Prediction (ratings, active_user=6,
                         near_user=Pearson.out$near_user_Pearson,
                         sim_x=Pearson.out$sim_Pearson,
                         KNN=length(Pearson.out$sim_Pearson))

ItemSelect (ratings, active_user=6, pre_x=predict)
```

`meanR.Results` *Average of results for all active users.*

Description

Average of MAE and elapsed time for all active users.

Usage

```
meanR.Results(obj_Results)
```

Arguments

`obj_Results`	An object of class `"Results"`.

Details

Due to the difference in the results of each active user, the average of all active users is calculated.

Value

An object of class `"meanR.Results"`, a list with components:

`call`	The call used.
`mean_MAE_Pearson`	Average of MAE obtained from the `"Pearson"` method for all active users.
`mean_MAE_NewKNN`	Average of MAE obtained from the `"NewKNN"` method for all active users.
`mean_MAE_Genetic`	Average of MAE obtained from the `"Genetic"` method for all active users.
`diff_MAE_GA_Pearson`	The difference of MAE in the `"Pearson"` method and `"Genetic"` algorithm.
`mean_Time_Pearson`	Average of the elapsed time of the `"Pearson"` method for all active users.
`mean_Time_NewKNN`	Average of the elapsed time of the `"NewKNN"` method for all active users.
`mean_Time_GA`	Average of the elapsed time of the `"Genetic"` method for all active users.

Examples

```
ratings <- matrix(c(  2,    5,  NaN,  NaN,  NaN,    4,
                    NaN,  NaN,  NaN,    1,  NaN,    5,
                    NaN,    4,    5,  NaN,    4,  NaN,
                      4,  NaN,  NaN,    5,  NaN,  NaN,
                      5,  NaN,    2,  NaN,  NaN,  NaN,
                    NaN,    1,  NaN,    4,    2,  NaN),nrow=6,byrow=TRUE)
```

```
active_users <- c(1:dim(ratings)[2])

Results.out <- Results(ratings, active_users, Threshold_KNN=4, max_scour=5,
                    min_scour=1, PopSize=100, MaxIteration=50,
                    CrossPercent=70, MutatPercent=20)

meanR.Results.out <- meanR.Results (obj_Results=Results.out)
```

NewKNN *Nearest Neighbors.*

Description

Determining of nearest neighbors and their id to determine the number of genes in a chromosome.

Usage

```
NewKNN(ratings, active_user, Threshold_KNN, max_scour, min_scour)
```

Arguments

ratings	A rating matrix whose rows are items and columns are users.
active_user	The id of an active user as an integer greater than zero (for example active_user<-6).
Threshold_KNN	Maximum number of neighbors.
max_scour	The maximum range of ratings.
min_scour	The minimum range of ratings.

Details

The number of neighbors for the active user determines the number of genes in the chromosome of the genetic algorithm. The fitness function is MAE which by being minimized, the similarity of the neighbor users is optimized within the processes of the genetic algorithm. The following equation is used to determine the starting points of the genetic algorithm, which are essentially approximation similarities. Using these starting points, the genetic algorithm converges faster.

$$sim_dif = (maxrating - dif)/sum(ratings)$$

$$rangeofdif : [minrating - 1, \ldots, maxrating - 1]$$

dif is the difference in the existing ratings. For example, for a difference of 0.5, the approximate similarity is $4.5/15$ and for a difference of 0, the similarity is $5/15$. In this method, the number of neighbors varies for each active user, so the problem of predetermining it is solved.

The steps of this function are:

1) The rating matrix is assigned to the form of the Item-user matrix (Items in rows and users in one column).

2) The users rating differences of each item are calculated for each pair of related users.

3) For each user, the related pairwise are separated from all rows in one column.

4) If a pairwise is repeated several times, the average values of the differences are calculated. The number of neighbor users is different for each active user.

5) The rating differences are sorted in ascending order.

6) Neighbor users are selected based on lower rating differences. If the threshold for the difference is already specified, the out-of-area relationships are eliminated.

Value

An object of class "NewKNN", a list with components:

call	The call used.
sim_NewKNN	The similarities between near users and the active user that have obtained from the "NewKNN" method.
pre_NewKNN	The predicted ratings for the active user by the NewKNN method.
item_NewKNN	A set of recommended items id, obtained from the NewKNN method.
near_user	Neighbors of the active user by the NewKNN method orderly.
time_NewKNN	The elapsed time in NewKNN method.

Author(s)

Farimah Houshmand Nanehkaran

Maintainer: Farimah Houshmand Nanehkaran <hoshmandcomputer@gmail.com>

References

Koohi, H. and Kiani, K. (2017). *A new method to find neighbor users that improves the performance of Collaborative Filtering*. Expert Systems with Applications, vol. 83, pp.30-39.

Examples

```
ratings <- matrix(c(  2,    5,  NaN,  NaN,  NaN,    4,
                    NaN,  NaN,  NaN,    1,  NaN,    5,
                    NaN,    4,    5,  NaN,    4,  NaN,
                      4,  NaN,  NaN,    5,  NaN,  NaN,
                      5,  NaN,    2,  NaN,  NaN,  NaN,
                    NaN,    1,  NaN,    4,    2,  NaN),nrow=6,byrow=TRUE)

NewKNN.out  <- NewKNN (ratings, active_user=6, Threshold_KNN=4,
                       max_scour=5, min_scour=1)
```

`Pearson` *Pearson method*

Description

The Pearson method is the most well-known method for finding users' similarity, so to compare the genetic-based method, the Pearson method has been implemented in this package.

Usage

```
Pearson(ratings, active_user, Threshold_KNN)
```

Arguments

`ratings`	A rating matrix whose rows are items and columns are users.
`active_user`	The id of an active user as an integer greater than zero (for example active_user<-6).
`Threshold_KNN`	Maximum number of neighbor users.

Details

Pearson Correlation Coefficient (PCC) is the similarity measure for Collaborative filtering recommender system, to evaluate how much two users are correlated [3].

Value

An object of class `"Pearson"`, a list with components:

`call`	The call used.
`sim_Pearson`	The similarity of the Pearson method.
`pre_Pearson`	The prediction of the Pearson method.
`item_Pearson`	A list of recommended items by the Pearson method.
`near_user_Pearson`	Neighbors of active user in the Pearson method orderly.
`time_Pearson`	The elapsed time of the Pearson method.

References

[1] Bobadilla, J., Ortega, F., Hernando, A. and Alcala, J. (2011). *Improving collaborative filtering recommender system results and performance using genetic algorithms*. Knowledge-based systems, vol. 24, no. 8, pp. 1310-1316.

[2] Lu, J., Wu, D., Mao, M., Wang W. and Zhang, G. (2015). *Recommender system application developments: a survey*. Decision Support Systems, vol. 74, pp. 12-32.

[3] Sheugh, L. and Alizadeh, S.H. (2015). *A note on pearson correlation coefficient as a metric of similarity in recommender system*. In 2015 AI & Robotics (IRANOPEN) (pp. 1-6). IEEE.

Examples

```
ratings <- matrix(c(  2,    5,  NaN,  NaN,  NaN,    4,
                    NaN,  NaN,  NaN,    1,  NaN,    5,
                    NaN,    4,    5,  NaN,    4,  NaN,
                      4,  NaN,  NaN,    5,  NaN,  NaN,
                      5,  NaN,    2,  NaN,  NaN,  NaN,
                    NaN,    1,  NaN,    4,    2,  NaN),nrow=6,byrow=TRUE)

Pearson.out  <- Pearson (ratings, active_user=6, Threshold_KNN=4)
```

`plotResults` *Methods for Results objects.*

Description

Provide standard methods for manipulating Results objects.

Usage

```
plotResults (active_users, obj_Results,
                        xlab = "Iteration", ylab = "MAE",
           main = "MAE (New KNN+GA) in CF Recommender Systems", ...)
```

Arguments

`active_users`	A vector of all active users id.
`obj_Results`	An object of class `"Results"`.
`xlab`, `ylab`, `main`	Graphics parameters.
`...`	Additional arguments passed on to the method.

Details

Methods for standard generic functions when dealing with objects of class `"Results"`

Value

a plot of the history of the process is produced with a `NULL` return value.

Examples

```
ratings <- matrix(c(  2,    5,  NaN,  NaN,  NaN,    4,
                    NaN,  NaN,  NaN,    1,  NaN,    5,
                    NaN,    4,    5,  NaN,    4,  NaN,
                      4,  NaN,  NaN,    5,  NaN,  NaN,
                      5,  NaN,    2,  NaN,  NaN,  NaN,
                    NaN,    1,  NaN,    4,    2,  NaN),nrow=6,byrow=TRUE)

active_users <- c(1:dim(ratings)[2])
```

```
Results.out <- Results(ratings, active_users, Threshold_KNN=4, max_scour=5,
                    min_scour=1, PopSize=100, MaxIteration=50,
                    CrossPercent=70, MutatPercent=20)

plotResults(active_users, Results.out, xlab = "Iteration", ylab = "MAE",
                main = "MAE (New KNN+GA) in CF Recommender Systems" )
```

`Prediction` *prediction function*

Description

Obtaining the ratings of items that not seen by the active user.

Usage

```
Prediction (ratings, active_user, near_user, sim_x, KNN)
```

Arguments

`ratings`	A rating matrix whose rows are items and columns are users.
`active_user`	The id of an active user as an integer greater than zero (for example active_user<-6).
`near_user`	Neighbor users.
`sim_x`	Similarity of neighbor users obtained from Similarity function.
`KNN`	The number of neighbor users that obtained for the active user from function or manually.

Details

The prediction formula is:

$$(p_x)^i = \bar{r}_x + ((\sum_{(} n \in nearusers)([sim(u_x, u_n).((r_n)^i - (\bar{r})_n)])) / (\sum_{(} n \in nearusers)(|sim(u_x, u_n)|)))$$

where $(P_x)^i$ is the prediction of the user x to an item i. $(\bar{r})_x$ is the average ratings of the user x and $\bar{r}_n$ is the average ratings of neighbors.

Value

`pre_y`	A set of predicted ratings for all items of the active user.

References

Moses, J.S. and Babu, L.D. (2018). *Evaluating Prediction Accuracy, Developmental Challenges, and Issues of Recommender Systems*. International Journal of Web Portals (IJWP), vol. 10, no. 2, pp. 61-79.

Singh, P., Ahuja, S. and Jain, S. (2019). *Latest Trends in Recommender Systems 2017*. In Advances in Data and Information Sciences, pp. 197-210. Springer, Singapore.

Examples

```
ratings <- matrix(c(  2,    5,  NaN,  NaN,  NaN,    4,
                    NaN,  NaN,  NaN,    1,  NaN,    5,
                    NaN,    4,    5,  NaN,    4,  NaN,
                      4,  NaN,  NaN,    5,  NaN,  NaN,
                      5,  NaN,    2,  NaN,  NaN,  NaN,
                    NaN,    1,  NaN,    4,    2,  NaN),nrow=6,byrow=TRUE)

Pearson.out  <- Pearson (ratings, active_user=6, Threshold_KNN=4)

predict <- Prediction (ratings, active_user=6,
                       near_user=Pearson.out$near_user_Pearson,
                       sim_x=Pearson.out$sim_Pearson,
                       KNN=length(Pearson.out$sim_Pearson))
```

`Results` *Results of all active users.*

Description

comparison of three methods (Genetic, NewKNN, Pearson) about MAE, elapsed time and predicted items.

Usage

```
Results(ratings, active_users, Threshold_KNN, max_scour, min_scour,
        PopSize=100, MaxIteration=50, CrossPercent=70, MutatPercent=20)
```

Arguments

`ratings`	A rating matrix whose rows are items and columns are users.
`active_users`	A vector of all active users id.
`Threshold_KNN`	Maximum number of neighbors.
`max_scour`	The maximum range of ratings.
`min_scour`	The minimum range of ratings.
`PopSize`	Population size (Number of chromosomes) in `Genetic` algorithm.
`MaxIteration`	Number of iterations in `Genetic` algorithm.

`CrossPercent`	Percentage of the `Genetic` algorithm population that participates in the Single-point crossover operator to generate new offspring.
`MutatPercent`	Percentage of the `Genetic` algorithm population that participates in the mutation.

Details

MAE is the average of the difference between real ratings of the active user and predicted ratings obtained from a method.

Value

An object of class `"Results"`, a list with components:

`call`	The call used.
`MAE_Pearson`	MAE obtained from the `"Pearson"` method.
`MAE_NewKNN`	MAE obtained from the `"NewKNN"` method.
`MAE_Genetic`	MAE obtained from the `"Genetic"` method.
`MAE_GA`	A vector of MAE in every `"Genetic"` iteration.
`time_Pearson`	Elapsed time of the `"Pearson"` method.
`time_NewKNN`	Elapsed time of the `"NewKNN"` method.
`time_Genetic`	Elapsed time of the `"Genetic"` method.

Note

MAE is abbreviate of Mean Absolute Error.

References

Salehi, M. (2014). *Latent feature based recommender system for learning materials using genetic algorithm*. Information Systems & Telecommunication, vol. 137.

See Also

`Genetic`, `NewKNN`, `Pearson`.

Examples

```
ratings<-matrix(c(  2,    5,  NaN,  NaN,  NaN,    4,
                  NaN,  NaN,  NaN,    1,  NaN,    5,
                  NaN,    4,    5,  NaN,    4,  NaN,
                    4,  NaN,  NaN,    5,  NaN,  NaN,
                    5,  NaN,    2,  NaN,  NaN,  NaN,
                  NaN,    1,  NaN,    4,    2,  NaN),nrow=6,byrow=TRUE)

active_users <- c(1:dim(ratings)[2])

Results.out <- Results(ratings, active_users, Threshold_KNN=4, max_scour=5,
                       min_scour=1, PopSize=100, MaxIteration=50,
                       CrossPercent=70, MutatPercent=20)
```

`Similarity_Pearson` *Similarity between users in Pearson method.*

Description

Finding the similarity with Pearson's method.

Usage

```
Similarity_Pearson(ratings, active_user, near_user)
```

Arguments

`ratings`	A rating matrix whose rows are items and columns are users.
`active_user`	The id of an active user as an integer greater than zero (for example active_user<-6).
`near_user`	A vector of neighbor users that in the Pearson method are all users.

Details

Obtaining the Pearson similarity through the cor command from state package does not produce the desired response.

Value

`sim_ac`	Pearson's similarity between the active user and all users.

References

Karabadji, N.E.I., Beldjoudi, S., Seridi, H., Aridhi, S. and Dhifli, W. (2018). *Improving memory-based user collaborative filtering with evolutionary multi-objective optimization*. Expert Systems with Applications, vol. 98, pp.153-165.

Examples

```
ratings <- matrix(c(  2,    5,  NaN,  NaN,  NaN,    4,
                    NaN,  NaN,  NaN,    1,  NaN,    5,
                    NaN,    4,    5,  NaN,    4,  NaN,
                      4,  NaN,  NaN,    5,  NaN,  NaN,
                      5,  NaN,    2,  NaN,  NaN,  NaN,
                    NaN,    1,  NaN,    4,    2,  NaN),nrow=6,byrow=TRUE)

sim.Pearson  <- Similarity_Pearson (ratings, active_user=6,
                                    near_user=c(1:dim(ratings)[2]))
```

Index