Package ‘lhcpackage’

February 10, 2015

Type Package

Title This project implements the local hill climbing method that was proposed by David Kane (Hutchin Hills), which tries to maximize the profit on a given n-dimensional landscape with various modes of connection.

Version 1.0

Date 2015-2-20

Author Michael Hyun Jae Lim

Maintainer Who to complain to `<m@m-ike.me>`

Description In an economic landscape, there are many local minima and maxima. While it is very hard to find the absolute maxima of the landscape, it is entirely possible to find many local maxima using the local hill climbing methods. The three methods entail: 1. steepest ascent, in which we look around the neighbors and move to the neighboring point with a maximum profit value, 2. median ascent in which we choose the median of the neighboring points with greater profit value, and 3. least ascent in which we choose the minimum profit value higher than the current profit value. It turns out that in many cases the least ascent has the most potential to optimize the given profit.

License GNU GENERAL PUBLIC LICENSE Version 2

R topics documented:

`chillclimb` *Compare Hill Climbing function*

Description

This function implements hill climbing algorithm by comparing various ascent methods and giving the result that has the optimal solution

Usage

```
chillclimb(X, Co, y)
```

Arguments

`X,Co,y`	This takes in the input of the vector for set of inputs, the randomly generated coefficients, and the mode of connection

Examples

```
chillclimb(X,Co,y)
```

`ghillclimb` *Hill Climbing Graphing function*

Description

This function implements hill climbing algorithm with various ascent methods, accompanying a graph

Usage

```
ghillclimb(X, Co, y, m)
```

Arguments

`X,Co,y,m`	This takes in the input of the vector for set of inputs, the randomly generated coefficients, the mode of connection, and the hill climbing mode (1= Steepest, 2= Median, 3= Least)

Examples

```
ghillclimb(X,Co,y,m)
```

`hillclimb` *Hill Climbing function*

Description

This function implements hill climbing algorithm with various ascent methods specified the user

Usage

```
hillclimb(X, Co, y, m)
```

Arguments

`X,Co,y,m`	This takes in the input of the vector for set of inputs, the randomly generated coefficients, the mode of connection, and the hill climbing mode (1= Steepest, 2= Median, 3= Least)

Examples

```
hillclimb(X,Co,y,m)
```

`lhcpackage` *This project implements the local hill climbing method that was proposed by David Kane (Hutchin Hills), which tries to maximize the profit on a given n-dimensional landscape with various modes of connection.*

Description

In an economic landscape, there are many local minima and maxima. While it is very hard to find the absolute maxima of the landscape, it is entirely possible to find many local maxima using the local hill climbing methods. The three methods entail: 1. steepest ascent, in which we look around the neighbors and move to the neighboring point with a maximum profit value, 2. median ascent in which we choose the median of the neighboring points with greater profit value, and 3. least ascent in which we choose the minimum profit value higher than the current profit value. It turns out that in many cases the least ascent has the most potential to optimize the given profit.

Details

Package:	lhc
Type:	Package
Version:	1.0
Date:	2015-2-20
License:	GNU GENERAL PUBLIC LICENSE Version 2

Suppose you have a set of inputs "X," and you want to find the local maximum with the coefficients given by the profit function "Pi(X)" (refer to Kane's paper on the nomenclature of the problem). We can use hillclimb(X,Co,y,m) function to complete the task. X denotes the set of inputs, Co denotes the set of coefficients, y denotes the number of allowed connections, and m denotes the method of ascent (1 = steepest ascent, 2 = median ascent, 3 = least ascent). The connections are defined by the neighborhoods, and a connection of mode "y" will link i-th and (i+y)-th coordinates of inputs together. Steepest ascent looks around the neighbors and move to the neighboring point with a maximum profit value. Median ascent chooses the median of the neighboring points with greater profit value. Least ascent we chooses the minimum profit value higher than the current profit value. Now, its sister function ghillclimb(X,Co,y,m) function completes the task, except it generates a graph that compares the relative profit value vs. the number of iterations. The function chillclimb(X,Co,y) compares the relative values for each of the hill climbing methods. Finally, thillclimb(y,No,Ev,St) generates a table that compares the hill climbing methods over many randomly chosen inputs and coefficients. y denotes the mode of connection once again, No denotes the number of inputs, Ev denotes the expected value for each of the inputs, and St denotes the number of sets that we want to include in the sample.

Author(s)

Michael Hyun Jae Lim

Maintainer: <m@m-ike.me>

References

"Local Hillclimbing on an Economic Landscape" -David Kane (SFI WORKING PAPER: 1996-08-065)

Examples

```
Suppose you have a
```

`neighbor_function` *Neighborhood function*

Description

This function generates all possible neighbors for the vector X, given in a matrix form

Usage

```
neighbor_function(X, y)
```

Arguments

`X`	This takes in the input of the vector for set of inputs, and the mode of connection (1st degree, 2nd degree, etc.)

Examples

```
neighbor_function(X)
```

`profit_function` *Profit function*

Description

This function generates the profit function for a random set of constants.

Usage

```
profit_function(X, Co)
```

Arguments

`X,Co`	This takes in the input of the vector for set of inputs and the randomly generated coefficients

Examples

```
profit_function(X,Co)
```

`rcoeff_function` *Random Coefficient function*

Description

This function generates a random set of constants

Usage

```
rcoeff_function(X)
```

Arguments

`X`	This takes in the input of the vector for set of inputs

Examples

```
rcoeff_function(X)
```

thillclimb *Table Hill Climbing function*

Description

This function implements hill climbing algorithm by comparing various ascent methods and giving the tables of solution

Usage

```
thillclimb(y, No, Ev, St)
```

Arguments

`y,No,Ev,St`	This takes in the mode of connection, number of inputs, expected value of each input, and the number of sets generated

Examples

```
thillclimb(y,No,Ev,St)
```

Index