Package 'trueskill'

August 29, 2016

Title Implementation the TrueSkill algorithm in R

Description An implementation of the TrueSkill algorithm (Herbrich, R., Minka, T. and Grapel, T) in R; a Bayesian skill rating system with inference by approximate message passing on a factor graph. Used by Xbox to rank gamers and identify appropriate matches. http://research.microsoft.com/en-us/projects/trueskill/default.aspx Current version allows for one player per team. Will update as time permits. Requires R version 3.0 as it is written with Reference Classes. URL: https://github.com/bhoung/trueskill-in-r Acknowledgements to Doug Zongker and Heungsub Lee for their python implementations of the algorithm and for the liberal reuse of Doug's code comments (@dougz and @sublee on github).

Version 0.1

Author Brendan Houng <brendan.houng@gmail.com>

Maintainer Brendan Houng <brendan.houng@gmail.com>

License GPL-3

Depends R (>= 3.0)

Imports methods

Collate 'factorgraph.r' 'init.r' 'competition.r' 'player.r'

NeedsCompilation no

Repository CRAN

Date/Publication 2013-05-22 08:00:08

R topics documented:

`trueskill-package` *Implementation of the TrueSkill algorithm*

Description

An R implementation of the TrueSkill Algorithm (Herbrich, R., Minka, T. and Grapel, T. [1]), a Bayesian skill rating system with inference by approximate message passing on a factor graph. Used by Xbox to rank gamers and identify appropriate matches.

http://research.microsoft.com/en-us/projects/trueskill/default.aspx

Current version allows for one player per team. Will update as time permits. Requires R version 3.0 as it is implemented with Reference Classes.

The code for the examples can be found at:

```
system.file('', package = 'trueskill')
```

Acknowledgements to Doug Zongker [2] and Heungsub Lee [3] for their python implementations of the algorithm and for the liberal reuse of Doug's code comments.

Details

Package:	trueskill
URL:	http://www.bhoung.com/trueskill
Version:	0.1
License:	Apache
Depends:	R (>= 3.0)
Built:	R 3.0.1

Main Functions and Classes

Reference Classes: `Gaussian`, `Player`, `Parameters`

Methods: `Multiply`, `Divide`

Functions: `AdjustPlayers`, `Trueskill`, `DrawMargin`, `DrawProbability`, `PrintList`

Data: `data`

Author(s)

Brendan Houng <brendan.houng@gmail.com>

References

[1] TrueSkill: A Bayesian Skill Rating System, Herbrich, R., Minka, T. and Grapel, T.

[2] Doug Zongker's python implementation:
https://github.com/dougz/trueskill

[3] Heungsub Lee's python implementation:
https://github.com/sublee/trueskill.

[4] Jeff Moser's explanatory notes:
http://www.moserware.com/2010/03/computing-your-skill.html

Examples

```
# Example 1.

# set default values for BETA, EPSILON and GAMMA where BETA is sigma / 2
# EPSILON is DrawProbability(0.1)
# GAMMA is sigma / 100
parameters <- Parameters$new()

Alice  <- Player(rank = 1, skill = Gaussian(mu = 25, sigma = 25 / 3), name = "1")
Bob    <- Player(rank = 2, skill = Gaussian(mu = 25, sigma = 25 / 3), name = "2")
Chris  <- Player(rank = 2, skill = Gaussian(mu = 25, sigma = 25 / 3), name = "3")
Darren <- Player(rank = 4, skill = Gaussian(mu = 25, sigma = 25 / 3), name = "4")

players <- list(Alice, Bob, Chris, Darren)

players <- AdjustPlayers(players, parameters)
PrintList(players)
print(Alice$skill)

# Relying on positional arguments looks much cleaner:
Alice  <- Player(1, Gaussian(25, 8.3), "Alice")
Bob    <- Player(2, Gaussian(25, 8.3), "Bob")
Chris  <- Player(2, Gaussian(25, 8.3), "Chris")
Darren <- Player(4, Gaussian(25, 8.3), "Darren")

# Example 2 - see https://gist.github.com/bhoung/5596282
# the example applies trueskill to tennis tournament data
# (runtime is approx 50 secs)
```

AdjustPlayers *Update the Skills of a List of Players*

Description

Runs the trueskill algorithm and updates the skills of a list of players. Assumes each team has one player.

Usage:

AdjustPlayers(players, parameters)

AdjustPlayers(list(Alice, Bob), parameters)

Arguments:

players is a list of player objects, for all the players who participated in a single game. A 'player object' is any object with a 'skill' field (a Gaussian) and a 'rank' field. Lower ranks are better; the lowest rank is the overall winner of the game. Equal ranks mean that the two players drew.

This function updates all the skills of the player objects to reflect the outcome of the game. Not the function sorts the players by rank and returns the sorted list. Creates all the variable nodes in the graph. "Teams" are each a single player; there's a one-to-one correspondence between players and teams.

parameters Parameters object to hold input variables:
beta, epsilon and gamma. See Parameters for more details.

Examples

```
Alice  <- Player(rank = 1, skill = Gaussian(mu = 25, sigma = 25 / 3), name = "1")
Bob    <- Player(rank = 2, skill = Gaussian(mu = 25, sigma = 25 / 3), name = "2")
Chris  <- Player(rank = 2, skill = Gaussian(mu = 25, sigma = 25 / 3), name = "3")
Darren <- Player(rank = 4, skill = Gaussian(mu = 25, sigma = 25 / 3), name = "4")

players <- list(Alice, Bob, Chris, Darren)

parameters <- Parameters$new()
players <- AdjustPlayers(players, parameters)
```

data *Australian Open data*

Description

A dataset containing tennis players and match outcomes from the 2012 Australian Open. Obtained from http://www.tennis-data.co.uk/data.php (2013).

Format

A data frame with 127 rows and 5 variables

Details

- Winner Player that won the game
- Loser Player that lost the game
- Round Round of the tournament
- WRank World Tennis Association Ranking of winning player
- LRank World Tennis Association Ranking of losing player

trueskill-package:

Refer to `trueskill-package` for links to related classes, functions and data.

`DrawMargin` *EPSILON or draw margin, used to set EPSILON in Parameters*

Description

Compute EPSILON or draw margin, a measure of how likely a draw is. Then pass to Parameters object.

Takes draw probability as input. Refer to `Parameters` for default input values.

Usage

```
DrawMargin(draw_probability = 0.10, beta = 25 / 6, total_players = 2)
```

Arguments

`draw_probability`	draw probability
`beta`	randomness in game
`total_players`	number of players

Examples

```
draw_margin <-DrawMargin(draw_probability = 0.10, beta = 25 / 6, total_players = 2)
parameters <- Parameters(beta = 25 / 6, epsilon = draw_margin, gamma = 25 / 300)
```

DrawProbability *Compute draw probability*

Description

Compute the draw probability given the draw margin (epsilon). Can be passed to DrawMargin to calculate EPSILON. Refer to Parameters for default input values.

Usage

```
DrawProbability(epsilon = 0.7404666, beta = 25 / 6, total_players = 2)
```

Arguments

epsilon	how common draws area
beta	randomness in game
total_players	number of players

Examples

```
draw_margin = 0.7404666
draw_probability = 0.10

draw_margin <- DrawMargin(draw_probability, beta = 25 / 6, total_players = 2)
draw_prob = DrawProbability(epsilon = draw_margin, beta = 25 / 6, total_players = 2)
```

Gaussian-class *Gaussian Class with args (mu, sigma) or (pi, tau)*

Description

Reference Class to create objects that represent normal distributions, which is how players' skills are represented. Gaussian takes arguments (mu, sigma) or (pi, tau), which default to (0, Inf) or (0, 0), respectively.

Note: for consistency reasons and not having to update two sets of values, the class only stores pi and tau. Therefore, g1$mu <- 25 and g1$sigma <- 8 does not work as expected, though g1$pi <- 0.04 and g1$tau <- 0.13 does.

Usage:

Gaussian(mu, sigma)

Gaussian(pi, tau)

Arguments:

mu is the mean skill

$$\mu$$

sigma is the std dev of skill

$$\sigma$$

pi is the precision

$$(1/\sigma^2)$$

tau is the precision adjusted mean

$$(\mu/\sigma^2)$$

Methods:

`* signature(e1 = Gaussian, e2 = Gaussian)`:...

`/ signature(e1 = Gaussian, e2 = Gaussian)`:...

`MuSigma()`: returns a list of c(mu, sigma)

`mu()`: returns mu

`sigma()`: returns sigma

See Also

`Divide` and `Multiply` These functions are not methods as a new copy of the Gaussian is produced (see example below).

Examples

```
g0 <- Gaussian(pi = 0.05, tau = 0.13)

g1 <- Gaussian(mu = 25, sigma = 8)
g2 <- Gaussian(30, 6)
g3 <- Multiply(g1, g2)

# these are equivalent
Gaussian$new(25, 8)
Gaussian$new(mu = 25, pi = 8)
Gaussian(mu = 25, pi = 8)
Gaussian(25, 8)

# approximately the same
Gaussian(pi = 0.016, tau = 0.391)
```

GaussianOperators *GaussianOperators*

Description

multiply or divide Gaussians equiavelent to the more convenient inline operator "*" or "/".

x * y

x / y

z <- x * y

z2 <- x / y

Usage

```
Multiply(x, y)
Divide(x, y)
```

Arguments

x	a gaussian object x
y	a gaussian object y

Examples

```
x <- Gaussian(25, 8)
y <- Gaussian(20, 6)
```

Parameters *Sets three parameters used in the TrueSkill algorithm.*

Description

Class creates an object to hold three parameters used in the TrueSkill algorithm. Passed to AdjustPlayer and Trueskill to perform calculations and update the Player objects or data.

The default parameters object is:

"Parameters [(beta, epilson, gamma)]: [(4.167, 0.74, 0.083)]"

where the default inputs are:

INITIAL_MU = 25.0

INITIAL_SIGMA = INITIAL_MU / 3.0

INITIAL_BETA = INITIAL_SIGMA / 2.0

INITIAL_GAMMA = INITIAL_SIGMA / 100.0

DRAW_PROBABILITY = 0.10

INITIAL_EPSILON = DrawMargin(DRAW_PROBABILITY, BETA)

Usage:

Parameters(beta, epsilon, gamma)

Arguments:

beta is a measure of how random the game is.

draw_probability probability that game ends in a draw. Can be calculated from DrawProbability function.

gamma is a small amount by which a player's uncertainty (sigma) is increased prior to the start of each game.

Examples

```
parameters <- Parameters()

# alternatively and equiavelently
draw_margin <-DrawMargin(draw_probability = 0.10, beta = 25 / 6, total_players = 2)
parameters <- Parameters(beta = 25 / 6, epsilon = draw_margin, gamma = 25 / 300)
```

Player *Player: class to hold the rank, skill and names of players*

Description

Reference class to create objects that represent players.

Usage:

Player(rank, skill, name)

Arguments:

rank rank of player in the match outcome

skill skill of player represented by Gaussian object e.g. Gaussian(mu = 25, sigma = 25/3)

name name the player for display purposes

Examples

```
Alice <- Player(1, Gaussian(25, 8), "Alice")
Bob <- Player(2, Gaussian(30, 7), "Bob")
players <- list(Alice, Bob)
PrintList(players)
```

PrintList *pretty print a list of players*

Description

pretty print a list of players (trivial function...)

Usage

```
PrintList(list)
```

Arguments

list	a list of player objects

Examples

```
Alice <- Player(1, Gaussian(25, 8), "Alice")
Bob <- Player(2, Gaussian(30, 7), "Bob")
players <- list(Alice, Bob)
PrintList(players)
```

Trueskill *Apply Trueskill to Tournament Data*

Description

Trueskill function to be applied to tournament data in dataframe format.

Data is required to be in long format with two rows for each match, one with player 1 first and one with player 2 first.

Matches should be sorted such that the second copy of the match appears in the second half of the dataframe.

The package currently only supports the trueskill algorithm with one player per team. Should this not match the data you are interested, a function could be written from AdjustPlayers, Player and Gaussian, and SetParameters.

An example is provided at https://gist.github.com/bhoung/5596282 (runtime is approx 50 secs).

Usage

```
Trueskill(data, parameters)
```

Arguments

data	a data frame (data) with columns: Player, Opponent, margin.
parameters	Parameters object to hold input variables.

Index