# Package ‘MultiGHQuad’

August 29, 2016

**Type** Package

**Title** Multidimensional Gauss-Hermite Quadrature

**Version** 1.2.0

**Date** 2016-08-11

**Description** Uses a transformed, rotated and optionally adapted n-dimensional grid of quadrature points to calculate the numerical integral of n multivariate normal distributed parameters.

**License** GPL-3

**Depends** R (>= 3.2.0), mvtnorm, Matrix

**Imports** fastGHQuad (>= 0.2)

**Suggests** testthat

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Author** Karel Kroeze [aut, cre],
Rivka de Vries [ctb],
Alwin Stegeman [ctb]

**Maintainer** Karel Kroeze <k.a.kroeze@utwente.nl>

**Repository** CRAN

**Date/Publication** 2016-08-12 12:47:57

## R topics documented:

MultiGHQuad-package *Multidimensional Gauss-Hermite Quadrature*

## Description

Functions to perform n-dimensional numerical integration on n parameters with a multivariate normal prior distribution.

## Details

Use `init.quad` to generate a quadrature grid, and `eval.quad` to evaluate the integral. Evaluation is performed with Gauss-Hermite quadrature, with a prior distribution that can be specified to any multivariate normal. Additionally, the grid can be adapted to any multivariate normal distribution - that is known to be close(r) to the posterior distribution under evaluation.

## Author(s)

Karel A Kroeze, <k.a.kroeze@utwente.nl>

## References

Jaeckel, P. (2005). *A note on multivariate Gauss-Hermite quadrature*. London: ABN-Amro. Retrieved from http://www.pjaeckel.webspace.virginmedia.com/ANoteOnMultivariateGaussHermiteQuadrature.pdf

Bock, R. D., & Mislevy, R. J. (1982). Adaptive EAP Estimation of Ability in a Microcomputer Environment. *Applied Psychological Measurement, 6*(4), 431-444. http://doi.org/10.1177/014662168200600405

## See Also

`init.quad`, `eval.quad`

eval.quad *Evaluation of multivariate normal distributed expectations*

## Description

Evaluates the posterior expectation of a (set of) parameters with a given likelihood function and multivariate normal prior distribution by Gauss-Hermite quadrature.

## Usage

```
eval.quad(FUN = function(x) 1, X = NULL, ..., W = NULL, forcePD = TRUE,
  debug = FALSE)
```

## Arguments

| | |
|---|---|
| FUN | log likelihood function of the parameters to be estimated. Defaults to `function(x) 1`, in which case only the prior likelihood is evaluated. |
| X | Matrix of quadrature points, see `init.quad`. Alternatively, the list of quadrature points and weights produced by `init.quad`. |
| ... | Additional arguments passed on to FUN. |
| W | Vector of weights, or `NULL` (the default) if provided by `X`. |
| forcePD | Logical, should the returned estimate be forced to the nearest positive definite matrix - if not already PD? If TRUE (Default: TRUE), `nearPD` is used to arrive at the closest PD matrix. |
| debug | Logical, should we return the results of FUN? |

## Details

The evaluated function is assumed to have a multivariate normal prior distribution, with a mean vector and covariance matrix specified in `init.quad`. The log-likelihood function defaults to an identity function `FUN(x) = 1`, which reduces the distribution under evaluation to specified prior distribution.

The integral under evaluation is;

$$\int_{-\infty}^{\infty} g(X|\mu,\Sigma) \times f(X) \times X dX$$

where $g(X|\mu,\Sigma)$ is the prior likelihood of X, and $f(X)$ is the likelihood of X in function `FUN`. If left default, the result is the expectation $E(X)$, where $X(\mu,\Sigma)$.

Note: FUN is evaluated in a loop, vectorisation is a future possibility. FUN must return a single scalar on the natural log-scale.

## Value

A vector with the evaluated integrals, with attribute `variance` containing the (co)variance (matrix) of the estimate(s), or the positive definite matrix closest to the estiamted covariance matrix.

## See Also

`init.quad` for creating quadrature points.

## Examples

```
### Basic example; E(X), X ~ N(0,1)
grid <- init.quad(Q = 1, prior = list(mu = 0, Sigma = diag(1)))
eval.quad(X = grid)

### Example; Rasch model person parameter
# E(theta), theta ~ N(0,1) * P(X = 1 | theta, beta), P is simplified rasch model
# set up rasch model with fixed beta, returns LL
rasch <- function(theta, beta, responses){
  p <- exp(theta - beta)/(1 + exp(theta - beta))
```

```
  q <- 1 - p
  return(log(p) * sum(responses == 1) + log(q) * sum(responses == 0))
}

# when theta == beta, P(X = 1) = .5, generate some bernoulli trials with p = .5
responses <- rbinom(5, 1, .5)

# get EAP estimate for theta, prior N(0,1)
eval.quad(rasch, grid, beta = 0, responses = responses)

# with more data, the estimate becomes more accurate, and variance decreases
eval.quad(rasch, grid, beta = 0, responses = rbinom(20, 1, .5))
eval.quad(rasch, grid, beta = 0, responses = rbinom(50, 1, .5))
eval.quad(rasch, grid, beta = 0, responses = rbinom(100, 1, .5))

### problem; the result starts to 'snap' to the closest quadrature point when
# the posterior distribution is too dissimilar to the prior.
evals <- eval.quad(rasch, grid, beta = 0, responses = rbinom(100, 1, .5), debug = TRUE)
evals.values <- attr(evals, "values")

# posterior density after 40 items
p <- plot(function(x) exp(dnorm(x, log = TRUE) +
                            rasch(x, beta = 1, responses = rbinom(100, 1, .5))),
          from = -3, to = 3)

# quadrature points used
points(grid$X, exp(grid$W)*max(p$y), pch = 20)

# the evaluation relies almost completely on one quadrature point,
# which causes results to 'snap' to that point.
# we could add more quadrature points...
grid2 <- init.quad(Q = 1, ip = 20)
points(grid2$X, exp(grid2$W)*max(p$y), pch = 20, col = "grey")

# but if the posterior is not centered on the prior, this quickly fails:
p <- plot(function(x) exp(dnorm(x, log = TRUE) +
                            rasch(x, beta = 2, responses = rbinom(100, 1, .5))),
          from = -3, to = 3)
points(grid2$X, exp(grid2$W)*max(p$y), pch = 20, col = "grey")

# additionally, adding extra quadrature points in a multidimensional
# problem quickly grows out of control.

### a better solution; adaptive quadrature grid.
# say we have an idea of where our parameter is located, through another estimator,
# or a previous estimate.
# we can then use this to adapt where our quadrature grid should be.
# get an estimate;
responses <- rbinom(10, 1, .5)
est <- eval.quad(rasch, grid, beta = 2, responses = responses)
print( est )

# adapt the grid;
```

```
grid3 <- init.quad(Q = 1, adapt = est)

# grid is now much closer to posterior
p <- plot(function(x) exp(dnorm(x, log = TRUE) +
                            rasch(x, beta = 2, responses = rep(c(0,1), each = 20))),
          from = -3, to = 3)
points(grid3$X, exp(grid3$W)*max(p$y), pch = 20, col = "grey")
est <- eval.quad(rasch, grid3, beta = 2, responses = responses)
print(est)
```

---

`init.quad` *Q-dimensional grid of quadrature points.*

---

## Description

Creates a flattened, rotated grid that incorporates correlation through an eigenvalue decomposition of the covariance matrix.

## Usage

```
init.quad(Q = 2, prior = list(mu = rep(0, Q), Sigma = diag(Q)),
  adapt = NULL, ip = 6, prune = FALSE, forcePD = FALSE, debug = FALSE)
```

## Arguments

| | |
|---|---|
| `Q` | Number of dimensions. Defaults to 2. Only required when `mu` and `Sigma` are not provided. |
| `prior` | List of prior mean `mu`, = `vector`, and covariance matrix `Sigma` = `matrix`, defaults to zero vector and identity matrix respectively. |
| `adapt` | List of adaptive mean `mu`, = `vector`, and covariance matrix `Sigma` = `matrix`, if `NULL` no adaptation is used. Defaults to NULL. |
| `ip` | Number of quadrature points *per dimension*. Defaults to 6. Note that the total number of quadrature points is `ip^Q`. |
| `prune` | Logical, should quadrature points with a very low weight be removed? Defaults to false. See details. |
| `forcePD` | Logical, should `adapt` and `prior` arguments be forced to the neares positive definite matrix - if not already PD? If TRUE (Default: FALSE), `nearPD` is used to arrive at the closest PD matrix. |
| `debug` | Logical, draws debugging plots when true. |

## Details

Creates a Q-dimensional grid by calling `expand.grid` on Q vectors of unidimensional quadrature points obtained with `gaussHermiteData`. The grid is then corrected for a prior distribution, and can optionally be adapted around a previous estimate. The resultant grid can be pruned to remove quadrature points that are unlikely to add information.

## Value

A list with a matrix `X` of `ip^Q` by `Q` quadrature points and a vector `W` of length `ip^Q` associated weights.

## See Also

`gaussHermiteData`, used to create unidimensional quadrature points, and `eval.quad` for evaluating the integral.

## Examples

```
### basic quadrature grid /w pruning.
mu <- c(0,0)
sigma <- matrix(c(1,.5,.5,1),2,2)
grid <- init.quad(Q = 2, prior = list(mu = mu, Sigma = sigma), ip = 10, prune = FALSE)
grid2 <- init.quad(Q = 2, prior = list(mu = mu, Sigma = sigma), ip = 10, prune = TRUE)
library(mvtnorm)
normal <- rmvnorm(1000, mu, sigma)
# noise
plot(normal, xlim = c(-6,6), ylim = c(-6,6), pch = 19, col = rgb(0,0,0,.5))
# full quad grid
points(grid$X, cex = exp(grid$W)/max(exp(grid$W))*4, col = 'red', pch = 20)
# pruned quad grid
points(grid2$X, cex = exp(grid2$W)/max(exp(grid2$W))*4, col = 'green', pch = 20)


### Adaptive quadrature grid
prior <- list(mu = c(0,0), Sigma = matrix(c(1,.5,.5,1),2,2))
adapt <- list(mu = c(-2,2), Sigma = prior$Sigma / 2)
grid <- init.quad(Q = 2, prior, ip = 10, prune = FALSE)
library(mvtnorm)
normal <- rmvnorm(1000, adapt$mu, adapt$Sigma)
# noise, centered at (-2, 2)
plot(normal, xlim = c(-6,6), ylim = c(-6,6), pch = 19, col = rgb(0,0,0,.5))
# initial quad grid, centered at (0, 0)
points(grid$X, cex = exp(grid$W)/max(exp(grid$W))*4, col = 'red', pch = 20)
# adapted grid
grid2 <- init.quad(Q =2, prior, adapt = adapt, ip = 10, prune = TRUE)
points(grid2$X, cex = exp(grid2$W)/max(exp(grid2$W))*4, col = 'green', pch = 20)
# the grid is adapted to the latest estimate, but weighted towards the prior
```

# Index