Package ‘Homework2’

December 5, 2013

Type Package

Title Mixture

Version 1.0

Date 2013-12-05

Author Stephen Cristiano

Maintainer `<scristia@jhsph.edu>`

Description Newton won’t converge

License GPL

R topics documented:

`mixture` *Mixture*

Description

Fit 2-component guassian mixture model using EM or newton (which doesn’t converge)

Usage

```
mixture(y, method, maxit = NULL, tol = 1e-08, param0 = NULL)
```

Arguments

`y`

`method`

`maxit`

`tol`

`param0`

Details

Newton won't converge on examples I tried.

Author(s)

Stephen Cristiano

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (y, method, maxit = NULL, tol = 1e-08, param0 = NULL)
{
    y <- unlist(y)
    method <- match.arg(method, choices = c("newton", "EM"))
    if (is.null(maxit)) {
        if (method == "newton")
            maxit <- 100
        else if (method == "EM")
            maxit <- 500
    }
    if (is.null(param0)) {
        v <- kmeans(y, 2, nstart = 10)
        mu0 <- sort(v$centers)
        s20 <- v$withinss/(v$size - 1)[order(v$centers)]
        nn <- v$size[order(v$centers)]
        lambda0 <- nn[1]/sum(nn)
        mu1 <- mu0[1]
        mu2 <- mu0[2]
        s1 <- s20[1]
        s2 <- s20[2]
        lambda <- lambda0
    }
    else {
        lambda <- param0[1]
        mu1 <- param0[2]
        mu2 <- param0[3]
        s1 <- param0[4]^2
        s2 <- param0[5]^2
    }
    n <- length(y)
    if (method == "newton") {
        lmix <- expression(-log((lambda/sqrt(s1)) * exp(-(y -
            mu1)^2/(s1)) + ((1 - lambda)/sqrt(s2)) * exp(-(y -
            mu2)^2/(s2))))
        dlmix <- deriv3(lmix, c("lambda", "mu1", "mu2", "s1",
            "s2"))
        mle.0 <- matrix(c(lambda, mu1, mu2, s1, s2), ncol = 1)
        itx <- 1
        repeat {
            ev <- eval(dlmix)
            grad <- colSums(attr(ev, "gradient"))
            hess <- colSums(attr(ev, "hessian"))
```

```
            b <- try(solve(hess), silent = TRUE)
            if (class(b) == "try-error")
                stop("Can't invert Hessian")
            mle.1 <- mle.0 + b %*% grad
            diff <- crossprod(mle.1 - mle.0)
            if (any(!is.finite(diff)))
                stop("non finite updates produced")
            itx = itx + 1
            if (diff < tol | itx > maxit)
                break
            lambda <- mle.1[1]
            mu1 <- mle.1[2]
            mu2 <- mle.1[3]
            s1 <- mle.1[4]
            s2 <- mle.1[5]
            mle.0 <- matrix(c(lambda, mu1, mu2, s1, s2), ncol = 1)
        }
        mle <- c(lambda = lambda, mu1 = mu1, mu2 = mu2, sigma1 = sqrt(s1),
            sigma2 = sqrt(s2))
        hess <- colSums(attr(ev, "hessian"))
        se <- sqrt(diag(solve(-hess)))
        stderr <- c(lambda = se[1], mu1 = se[2], mu2 = se[3],
            sigma1 = se[4], sigma2 = se[5])
        return(list(mle = mle, stderr = stderr))
    }
    if (method == "EM") {
        prev <- c(lambda, mu1, mu2, s1, s2)
        itx <- 1
        repeat {
            z.tilde <- lambda * dnorm(y, mu1, sqrt(s1))/(lambda *
                dnorm(y, mu1, sqrt(s1)) + (1 - lambda) * dnorm(y,
                mu2, sqrt(s2)))
            mu1 <- sum(z.tilde * y)/sum(z.tilde)
            mu2 <- sum((1 - z.tilde) * y)/sum(1 - z.tilde)
            s1 <- sum(z.tilde * (y - mu1)^2)/sum(z.tilde)
            s2 <- sum((1 - z.tilde) * (y - mu2)^2)/sum(1 - z.tilde)
            lambda <- sum(z.tilde)/n
            diff <- crossprod(c(lambda, mu1, mu2, s1, s2) - prev)
            itx <- 1
            if (diff < tol | itx > maxit)
                break
            prev <- c(lambda, mu1, mu2, s1, s2)
        }
        exp.score <- matrix(NA, n, 5)
        exp.score[, 1] <- z.tilde/lambda - (1 - z.tilde)/(1 -
            lambda)
        exp.score[, 2] <- z.tilde * (y - mu1)/s1
        exp.score[, 3] <- (1 - z.tilde) * (y - mu2)/s2
        exp.score[, 4] <- (-1/sqrt(s1) + 1/sqrt(s1)^3 * (y -
            mu1)^2) * z.tilde
        exp.score[, 5] <- (-1/sqrt(s2) + 1/sqrt(s2)^3 * (y -
            mu2)^2) * (1 - z.tilde)
        cprd.score <- crossprod(exp.score)
        se <- sqrt(diag(solve(cprd.score)))
        mle <- c(lambda = lambda, mu1 = mu1, mu2 = mu2, sigma1 = sqrt(s1),
            sigma2 = sqrt(s2))
        stderr <- c(lambda = se[1], mu1 = se[2], mu2 = se[3],
```

```
            sigma1 = se[4], sigma2 = se[5])
        return(list(mle = mle, stderr = stderr))
    }
}
```

Index