```r
log.app <- function(Xapp, zapp, intr, epsi) {
    n <- dim(Xapp)[1]
    p <- dim(Xapp)[2]

    Xapp <- as.matrix(Xapp)

    if (intr == T) { # on ajoute une ordonnée à l'origine
        Xapp <- cbind(rep(1,n),Xapp)
        p <- p + 1
    }

    targ <- matrix(as.numeric(zapp),nrow=n) # ti: la réalisation d'une variable Ti~B(pi)
    targ[which(targ==2),] <- 0              # remplacer la classe 2 par 0
    Xapp_transposed <- t(Xapp)

    beta <- matrix(0,nrow=p,ncol=1)

    conv <- F
    iter <- 0
    while (conv == F) {
        iter <- iter + 1
        beta_old <- beta

        prob_w1 <- postprob(beta, Xapp) # P(w1|x)
        prob_w2 <- 1 - prob_w1
        MatW <- diag(as.numeric(prob_w1 * prob_w2)) # W: Wii = pi(1-pi)

        mat_hessienne <- -Xapp_transposed %*% MatW %*% Xapp
        mat_hessienne_inverse <- solve(mat_hessienne)
        gradient_w1 <- Xapp_transposed %*% (targ - prob_w1)
        beta <- beta_old - (mat_hessienne_inverse %*% gradient_w1)

        if (norm(beta - beta_old) < epsi) {
            conv <- T
        }
    }

    prob_w1 <- postprob(beta, Xapp) # P(w1|x)
    prob_w2 <- 1 - prob_w1
    out <- NULL
    out$beta <- beta
    out$iter <- iter
    out$logL <- sum(targ*prob_w1+(1-targ)*(prob_w2))
    out
}
```

```r

log.val <- function(beta, Xtst) {
    m <- dim(Xtst)[1]
    p <- dim(beta)[1]
    pX <- dim(Xtst)[2]

    Xtst <- as.matrix(Xtst)

    if (pX == (p-1))
    {
        Xtst  <- cbind(rep(1,m),Xtst)
    }

    prob_w1 <- postprob(beta, Xtst) # P(w1|x)
    prob_w2 <- 1 - prob_w1
    prob <- cbind(prob_w1, prob_w2)
    pred <- max.col(prob)

    out <- NULL
    out$prob <- prob
    out$pred <- pred
    return(out)
}


# calculer des probabilités a posteriori de la classe 1
postprob <- function(beta, X) {
    X <- as.matrix(X)

    prob <- t(exp(t(beta)%*%t(X))/(1+exp(t(beta)%*%t(X))))
}
```