The EM algorithm is a nice tool for maximizing non-standard likelihood functions, particularly in cases of "missing data." The results are shown below.

We implement EM algorithm for probit as mentioned below. Then calculate EM and GLM(family=binomial(link="probit")) for different initial values. The results for coefficient estimation and number of iteration are shown below. The main challenge with EM is that it's convergence may be slow.

1. E-Step: compute

$$Z^{(t+1)} = \mathbb{E}\left[Z|y, \beta^{(t)}\right],$$

$$Z_i^{(t+1)} = \begin{cases} x_i^T\beta^{(t)} - \frac{\phi(x_i^T\beta^{(t)})}{\Phi(-x_i^T\beta^{(t)})} & \text{if } y_i = 0 \\ x_i^T\beta^{(t)} + \frac{\phi(x_i^T\beta^{(t)})}{1-\Phi\left(-x_i^T\beta^{(t)}\right)} & \text{if } y_i = 1 \end{cases}.$$

2. M-Step:

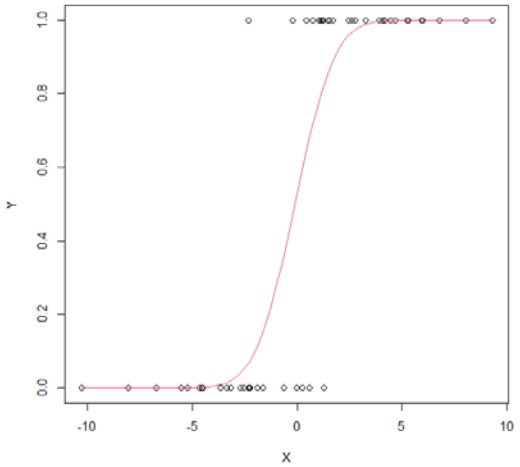
$$\beta^{(t+1)} = (X^TX)^{-1}X^TZ^{(t+1)},$$

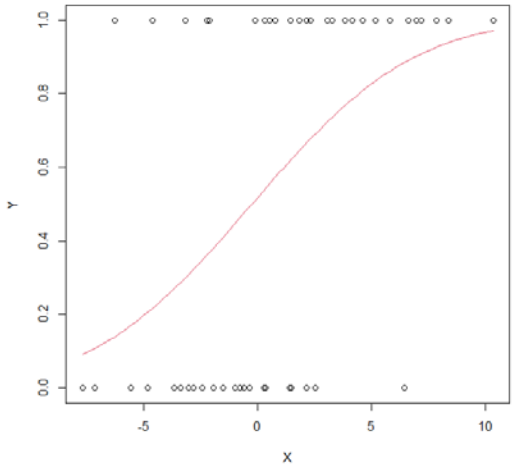
3. Continue until convergence

theta = (1,3)			
Model	iter	intercept	x
GLM	8	0.0705	0.6739
EM	700	0.0705	0.6739

theta=(0.1, 0.2)			
Model	iter	intercept	x
GLM	4	4382619.0000	0.1792
EM	41	0.0438	0.1792

theta = (50, 20)			
Model	iter	intercept	x
GLM	didn't converge	57.3545	23.2981
EM	didn't converge	49.9984	20.0003





Appendix: R code

```
# probit regression via EM

# Y for complete data, X for the binary response and U for the predictor variable. .

em.probreg <- function(U, X, theta0) {

  maxit <- 1000

  tol <- 1e-10

  i <- 0

  M <- solve(t(U) %*% U) %*% t(U)

  theta <- theta0

  m <- U %*% theta

  repeat {

    i <- i + 1

    v <- (2 * X - 1) * dnorm(m) / pnorm((2 * X - 1) * m)

    up <- M %*% v

    theta <- theta + up

    if(max(abs(up)) < tol || i >= maxit) break

    m <- U %*% theta

  }

  return(list(iterations=i, estimate=as.numeric(theta)))

}


A <- 0

B <- 1

x <- rnorm(50, 0, 4)

y <- rbinom(50, 1, pnorm(A + B * x))
```

```
nloglik <- function(theta) {


  a <- theta[1]

  b <- theta[2]

  p <- pnorm(a + b * x)

  o <- sum(dbinom(y, size=1, prob=p, log=TRUE))

  return(-o)

}

theta0 <- c(1,3)

o.optim <- optim(theta0, nloglik, method="BFGS"); print(o.optim)

glm1 <- glm(y ~ x, family=binomial(link="probit")); print(list(glm1$coef, glm1$iter))

o.em <- em.probreg(as.matrix(cbind(rep(1, 50), x)), y, theta0); print(o.em)

plot(x, y, xlab="X", ylab="Y")

f <- function(x) pnorm(o.em$estimate[1] + o.em$estimate[2] * x)

curve(f, add=TRUE, col=2)
```