Newton's method and Scoring

## 1. Maximizing likelihood functions using Newton's method (1-dim.)

을 어떤 모집단으로부터의 랜덤샘플이라 하고, 를 확률밀도(질량)함수라 하자. 여기서 는 모수이다.

최대우도추정량(MLE) 은 다음과 같은 log-likelihood를 최대화하는 값으로 정의된다.

적절한 조건 하에서 MLE를 찾는 것은 다음과 같은 우도방정식을 푸는 것으로 귀결된다.

따라서, 우도방정식의 해를 구하기 위한 Newton's algorithm은 다음과 같이 표현된다.

#### 1.1 Observed vs expected information

피셔정보행렬(Fisher information matrix)의 정의를 떠올려 보아라.

이 때, 를 observed information이라 하고, 그 기대값 을 expected (Fisher) information 이라 한다.

* 는 양수의 값을 가져야 한다. (why?)
* 는 최대값 근처에서 우도함수의 모양(특히, 뾰족한 정도)을 결정한다.
* 의 값이 모든 에 대하여 양수일 경우 Newton's method에 의해 생성되는 수열의 MLE로의 수렴성이 안정적이다. 하지만, 경우에 따라 음수인 영역이 존재할 수 있으며 이 경우 알고리즘이 불안정할 수 있다.

#### 1.2 Example

##### 1.2.1 성공확률

이 로부터의 랜덤샘플일 때, 성공확률 에 대한 MLE를 계산해보자.

1. 으로부터 임을 알 수 있다.
2. Newton's method에 의해

p <- 0.1  
g <- function(x,p){sum(x)/p - (length(x)-sum(x))/(1-p)}  
dg <- function(x,p){-sum(x)/p^2 - (length(x)-sum(x))/(1-p)^2}  
tol <- 10^{-8}  
set.seed(1)  
x=rbinom(100,1,0.3)  
for (i in 1:100)  
 {  
 J <- dg(x,p)  
 newp <- p - g(x,p)/dg(x,p)  
 err <- abs(newp-p)  
 print(c(i,p))  
 if (err<tol) break  
 p<-newp  
 }

## [1] 1.0 0.1  
## [1] 2.0000000 0.1744361  
## [1] 3.0000000 0.2622224  
## [1] 4.0000000 0.3128146  
## [1] 5.0000000 0.3199112  
## [1] 6.00 0.32  
## [1] 7.00 0.32

c(newp,sum(x)/length(x))

## [1] 0.32 0.32

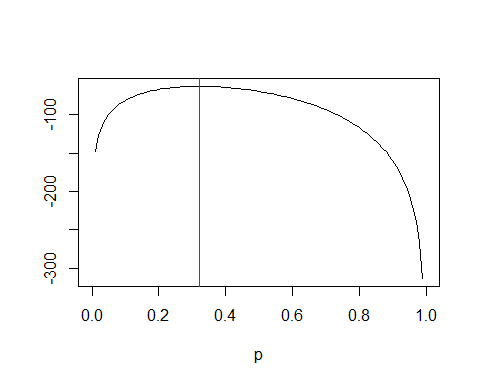
g(x,p)

## [1] 1.278977e-13

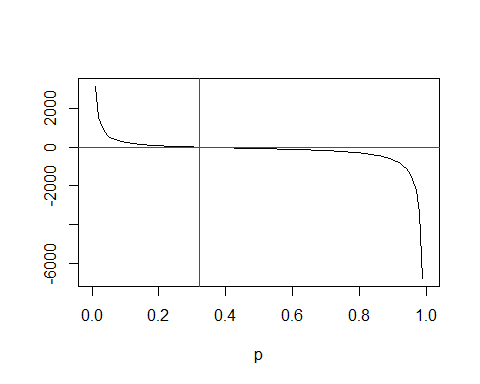
dg(x,p)

## [1] -459.5588

plot(seq(0,1,0.01),sum(x)\*log(seq(0,1,0.01)) + (length(x)-sum(x))\*log(1-seq(0,1,0.01)),xlab="p",ylab="",type="l")  
abline(v=newp,col="red")



plot(seq(0,1,0.01),g(x,seq(0,1,0.01)),xlab="p",ylab="",type="l")  
abline(h=0,v=newp,col="red")



##### 1.2.2 로지스틱(logistics) 분포

이 다음과 같은 확률분포로부터의 랜덤샘플일 때, 모수 에 대한 MLE를 계산해보자.

cf) .

dl <- function(x,theta){length(x)-2\*sum( exp(-x+theta)/(1+exp(-x+theta)))}  
ddl <- function(x,theta){-2\*sum( exp(-x+theta)/(1+exp(-x+theta))^2)}  
tol <- 10^{-8}  
set.seed(1)  
x=rlogis(100,5,1)  
theta <- mean(x)  
for (i in 1:100)  
 {  
 newtheta <- theta - dl(x,theta)/ddl(x,theta)  
 err <- abs(newtheta-theta)  
 print(c(i,theta))  
 if (err<tol) break  
 theta<-newtheta  
 }

## [1] 1.000000 5.087336  
## [1] 2.000000 5.099729  
## [1] 3.000000 5.099729

dl(x,theta)

## [1] -2.842171e-14

## 2. Newton's method

* Example (syetem of equations)

의 해를 구하여라.

위 문제는 로 정리 가능함.

앞 장에서 함수가 단변량일 때는 각 지점 에서의 1차 테일러 근사를 이용하여 수열 을 생성하였다. 비슷한 방식으로 접근하되 함수가 다변량이므로 다변량함수에 대한 테일러 근사를 이용하여야 한다.

* 이변량함수에 대한 근방에서의 1차 테일러 전개

cf) 는 를 나타냄.

* 로 두고 에서 테일러 근사를 각 함수에 대해서 적용하여 보면

따라서,

즉, 가 한번 업데이트된 해의 근사치이다.

* 이변량의 경우 일반적으로 Newton's method를 정리하면,

위 example의 경우에 적용하면,

z <- c(1,0)  
g1 <- function(z){z[1]^2+z[2]^2-2}  
g2 <- function(z){z[1]-z[2]-2}  
dg11<- function(z){2\*z[1]}  
dg12<- function(z){2\*z[2]}  
dg21<- function(z){1}  
dg22<- function(z){-1}  
tol <- 10^{-4}  
for (i in 1:100)  
 {  
 J <- matrix(c(dg11(z),dg21(z),dg12(z),dg22(z)),2,2)  
 newz <- z - solve(J)%\*%c(g1(z),g2(z))  
 err <- max(abs(newz-z))  
 print(c(i,z))  
 if (err<tol) break  
 z<-newz  
 }

## [1] 1 1 0  
## [1] 2.0 1.5 -0.5  
## [1] 3.00 1.25 -0.75  
## [1] 4.000 1.125 -0.875  
## [1] 5.0000 1.0625 -0.9375  
## [1] 6.00000 1.03125 -0.96875  
## [1] 7.000000 1.015625 -0.984375  
## [1] 8.0000000 1.0078125 -0.9921875  
## [1] 9.0000000 1.0039062 -0.9960938  
## [1] 10.0000000 1.0019531 -0.9980469  
## [1] 11.0000000 1.0009766 -0.9990234  
## [1] 12.0000000 1.0004883 -0.9995117  
## [1] 13.0000000 1.0002441 -0.9997559  
## [1] 14.0000000 1.0001221 -0.9998779

newz

## [,1]  
## [1,] 1.000061  
## [2,] -0.999939

c(g1(newz),g2(newz))

## [1] 7.450581e-09 0.000000e+00

* 일반적으로 개의 서로 다른 -dimensional 함수에 대한 연립방정식을 푸는 Newton's method

단,

, ,

* Example

이 언제 최소값을 가지는가?

위 함수의 최소값을 찾는 문제는 을 푸는 것과 동치임. 그러므로 여기에 Newton's method를 적용하는 것으로 충분함.

z <- c(1,1)  
g1 <- function(z){4\*z[1]^3+4\*z[1]^2\*z[2]}  
g2 <- function(z){4\*z[2]^3+4\*z[2]^2\*z[1]}  
dg11<- function(z){12\*z[1]^2+4\*z[2]^2}  
dg12<- function(z){8\*z[1]\*z[2]}  
dg21<- function(z){8\*z[1]\*z[2]}  
dg22<- function(z){12\*z[2]^2+4\*z[1]^2}  
tol <- 10^{-4}  
for (i in 1:100)  
 {  
 J <- matrix(c(dg11(z),dg21(z),dg12(z),dg22(z)),2,2)  
 newz <- z - solve(J)%\*%c(g1(z),g2(z))  
 err <- max(abs(newz-z))  
 print(c(i,z))  
 if (err<tol) break  
 z<-newz  
 }

## [1] 1 1 1  
## [1] 2.0000000 0.6666667 0.6666667  
## [1] 3.0000000 0.4444444 0.4444444  
## [1] 4.0000000 0.2962963 0.2962963  
## [1] 5.0000000 0.1975309 0.1975309  
## [1] 6.0000000 0.1316872 0.1316872  
## [1] 7.0000000 0.0877915 0.0877915  
## [1] 8.00000000 0.05852766 0.05852766  
## [1] 9.00000000 0.03901844 0.03901844  
## [1] 10.00000000 0.02601229 0.02601229  
## [1] 11.00000000 0.01734153 0.01734153  
## [1] 12.00000000 0.01156102 0.01156102  
## [1] 13.000000000 0.007707347 0.007707347  
## [1] 14.000000000 0.005138231 0.005138231  
## [1] 15.000000000 0.003425487 0.003425487  
## [1] 16.000000000 0.002283658 0.002283658  
## [1] 17.000000000 0.001522439 0.001522439  
## [1] 18.000000000 0.001014959 0.001014959  
## [1] 1.900000e+01 6.766395e-04 6.766395e-04  
## [1] 2.00000e+01 4.51093e-04 4.51093e-04  
## [1] 2.100000e+01 3.007287e-04 3.007287e-04  
## [1] 2.200000e+01 2.004858e-04 2.004858e-04

newz

## [,1]  
## [1,] 0.0001336572  
## [2,] 0.0001336572

c(g1(newz),g2(newz))

## [1] 1.910147e-11 1.910147e-11

cf) 업데이트 되는 알고리즘을 자세하게 써서 정리하여 보아라.

## 3. Maximizing likelihood functions Using Newton's method (-dim.)

을 어떤 모집단으로부터의 랜덤샘플이라 하고, 를 확률밀도(질량)함수라 하자. 여기서 는 모수이다.

최대우도추정량(MLE) 은 다음과 같은 log-likelihood를 최대화하는 값으로 정의된다.

적절한 조건 하에서 MLE를 찾는 것은 다음과 같은 우도방정식을 푸는 것으로 귀결된다.

의 1차, 2차 (편)미분함수들을 다음과 같이 정의하면,

우도방정식의 해를 구하기 위한 Newton's algorithm은 다음과 같이 표현된다.

여기서 를 의 gradient vector, 를 Hessian matrix 라 한다.

#### 3.1 Observed vs expected information

피셔정보행렬(Fisher information matrix)의 정의를 떠올려 보아라.

여기서 는 Hessian matrix이다. 즉,

임을 알 수 있다. 이 때, 를 observed information이라 하고, 그 기대값을 expected (Fisher) information 이라 한다.

* 는 최대값 근처에서 우도함수의 모양(특히, 뾰족한 정도)을 결정한다.
* 의 값이 모든 에 대하여 양정치행렬(positive definite)인 경우 Newton's method에 의해 생성되는 수열의 MLE로의 수렴성이 안정적이다. 하지만, 경우에 따라 그렇지 않을 수 있으며 이 경우 알고리즘이 불안정할 수 있다.

#### 3.2 Example

이 로부터의 랜덤샘플일 때, 에 대한 MLE.

theta <- c(1,1)  
dl1 <- function(x,m,s){sum(x-m)/s}  
dl2 <- function(x,m,s){sum((x-m)^2)/(2\*s^2)-length(x)/(2\*s)}  
ddl11<- function(x,m,s){-length(x)/s}  
ddl12<- function(x,m,s){-sum(x-m)/(s^2)}  
ddl21<- function(x,m,s){-sum(x-m)/(s^2)}  
ddl22<- function(x,m,s){-sum((x-m)^2)/(s^3)+length(x)/(2\*s^2)}  
tol <- 10^{-8}  
set.seed(1)  
x <- rnorm(100,2,3)  
for (i in 1:100)  
 {  
 ddl <- matrix(c(ddl11(x,theta[1],theta[2]),ddl21(x,theta[1],theta[2]),ddl12(x,theta[1],theta[2]),ddl22(x,theta[1],theta[2])),2,2)  
 newtheta <- theta - solve(ddl)%\*%c(dl1(x,theta[1],theta[2]),dl2(x,theta[1],theta[2]))  
 err <- sum((newtheta-theta)^2)  
 print(c(i,theta))  
 if (err<tol) break  
 theta<-newtheta  
 }

## [1] 1 1 1  
## [1] 2.000000 1.887478 1.331045  
## [1] 3.000000 2.135969 1.908983  
## [1] 4.000000 2.246471 2.711758  
## [1] 5.000000 2.295932 3.750934  
## [1] 6.000000 2.316724 4.964009  
## [1] 7.000000 2.324314 6.136989  
## [1] 8.000000 2.326362 6.919990  
## [1] 9.000000 2.326651 7.168948  
## [1] 10.000000 2.326662 7.188147  
## [1] 11.000000 2.326662 7.188250

newtheta

## [,1]  
## [1,] 2.326662  
## [2,] 7.188250

## 4. Scoring method

앞에서 이야기한 바대로 Newton's method는 빠른 수렴속도를 보장하는 대신 알고리즘의 불안정성 문제가 발생하는 경우가 빈번하다. 또한, Hessian matrix를 매 iteration step마다 계산하고 inverse를 취하여야 하는데 이 계산과정에 매우 큰 자원(혹은 시간)이 소요되는 경우가 있다. 이러한 단점을 보완하기 위하여 제안된 방법이 Scoring method이다. 이 방법은 기본적인 알고리즘의 형태는 유지하되 observed information인 (- Hessian matrix)를 그 기대값인 expected information (피셔정보행렬)로 대체하는 것이다.

* 는 에 비해 계산이 간단한 경우가 많다.
* 는 언제나 양수의 값(모수의 차원이 1인 경우)을 가지거나 양정치행렬(positive (semi-)definite matrix)이다. (why?)

위와 같은 이유에 의해 알고리즘이 Newton's method에 비해 안정화되는 경향이 있으며, 계산량도 더 적어진다.

* cf1) 어떤 경우에는 Newton's method와 scoring method가 동일하다.
* cf2) 피셔정보행렬로 대체하는 것이 가장 일반적이나, 다른 방법들도 존재한다. (예를 들어 항등행렬 로 대체)
* cf3) 피셔정보행렬의 역행렬(을 true parameter에서 계산한 값)이 MLE의 점근적분산인 것으로 알려져 있고, Scoring method는 이 값을 자동으로 제공한다.

#### 4.1 성공확률 (revisited)

이 로부터의 랜덤샘플일 때, 성공확률 에 대한 MLE.

p <- 0.1  
g <- function(x,p){sum(x)/p - (length(x)-sum(x))/(1-p)}  
dg <- function(x,p){-length(x)/(p\*(1-p))}  
tol <- 10^{-8}  
set.seed(1)  
x=rbinom(100,1,0.3)  
for (i in 1:100)  
 {  
 J <- dg(x,p)  
 newp <- p - g(x,p)/dg(x,p)  
 err <- abs(newp-p)  
 print(c(i,p))  
 if (err<tol) break  
 p<-newp  
 }

## [1] 1.0 0.1  
## [1] 2.00 0.32

c(newp,sum(x)/length(x))

## [1] 0.32 0.32

cf) 알고리즘을 구체적으로 작성하여 보아라.

#### 4.2 로지스틱(logistics) 분포 (revisited)

이 다음과 같은 확률분포로부터의 랜덤샘플일 때, 모수 에 대한 MLE를 계산해보자.

dl <- function(x,theta){length(x)-2\*sum( exp(-x+theta)/(1+exp(-x+theta)))}  
ddl <- function(x,theta){-2\*sum( exp(-x+theta)/(1+exp(-x+theta))^2)}  
tol <- 10^{-8}  
set.seed(1)  
x=rlogis(100,5,1)  
theta <- mean(x)  
for (i in 1:100)  
 {  
 newtheta <- theta - dl(x,theta)/(-length(x)/3)  
 err <- abs(newtheta-theta)  
 print(c(i,theta))  
 if (err<tol) break  
 theta<-newtheta  
 }

## [1] 1.000000 5.087336  
## [1] 2.000000 5.100649  
## [1] 3.00000 5.09966  
## [1] 4.000000 5.099734  
## [1] 5.000000 5.099728  
## [1] 6.000000 5.099729  
## [1] 7.000000 5.099729

dl(x,theta)

## [1] 7.471326e-08

cf) expected information의 계산이 너무 복잡한 경우 Scoring method의 적용이 더 어려운 경우도 있다.

#### 4.3 정규분포 (revisited)

으로부터

theta <- c(1,1)  
dl1 <- function(x,m,s){sum(x-m)/s}  
dl2 <- function(x,m,s){sum((x-m)^2)/(2\*s^2)-length(x)/(2\*s)}  
ddl11<- function(x,m,s){length(x)/s}  
ddl22<- function(x,m,s){length(x)/(2\*s^2)}  
tol <- 10^{-8}  
set.seed(1)  
x <- rnorm(100,2,3)  
for (i in 1:100)  
 {  
 ddl <- matrix(c(ddl11(x,theta[1],theta[2]),0,0,ddl22(x,theta[1],theta[2])),2,2)  
 newtheta <- theta + solve(ddl)%\*%c(dl1(x,theta[1],theta[2]),dl2(x,theta[1],theta[2]))  
 err <- sum((newtheta-theta)^2)  
 print(c(i,theta))  
 if (err<tol) break  
 theta<-newtheta  
 }

## [1] 1 1 1  
## [1] 2.000000 2.326662 8.948283  
## [1] 3.000000 2.326662 7.188250

newtheta

## [,1]  
## [1,] 2.326662  
## [2,] 7.188250

## 5. Exercise

1. Weibull distribution
2. Cauchy distribution
3. Gamma distribution
4. Geometric distribution (with re-parametrization)

## 6. Nonlinear regression

#### 6.1 Normally distributed errors

이 로부터의 랜덤샘플이라고 하자. 추정모수는 이고 는 알려져 있는 상수일 때, 우도함수는 다음과 같이 표현된다.

따라서,

그러므로, Scoring method를 적용하면,

이때, 에 대한 업데이트 알고리즘을 Gauss-Newton 알고리즘이라고 부른다.

위 알고리즘에서 특이한 점은 의 업데이트에 은 전혀 관여하지 않는다는 것이다. 그리고, 에 대한 추정량은 의 최종 업데이트 값을 위에 대입함으로써 얻어진다.

#### 6.2 Least squares

이 평균이 인 분포로부터의 랜덤샘플이라고 할 때, 제곱합을 다음과 같이 정의.

를 최소로 하는 를 Newton's method에 의해 찾으면,

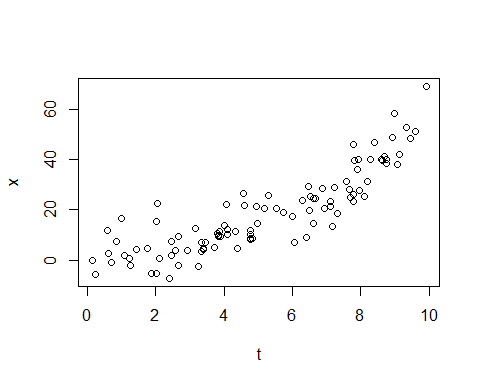
Hessian에서 부분은 종종 생략되고 (가 선형함수에 가깝거나 오차를 나타내는 가 충분히 작으면 생략하는 것에 큰 무리가 없음.) 첫번째 term만 남기기도 한다. 이렇게 하면, 6.1의 Gauss-Newton 알고리즘과 같아진다. 또한, 그렇게 되면 근사적인 Hessian matrix가 양정치행렬이 되어 알고리즘이 안정화된다 (MLE를 얻기 위해 scoring method를 슬 때 나타나는 장점과 같다.). 이렇게 구한 추정량을 라 하면,

1. 의 점근적 분포는 , .
2. 은 으로 추정될 수 있다.
3. 는 에 의해 추정될 수 있다.

* Exponential population model

where .

ddmu11 <- function(t,phi){sum(exp(2\*phi[2]\*t))}  
ddmu12 <- ddmu21 <- function(t,phi){sum( phi[1]\*t\*exp(2\*phi[2]\*t) )}  
ddmu22 <- function(t,phi){sum( phi[1]^2\*t^2\*exp(2\*phi[2]\*t) )}  
dmu1 <- function(x,t,phi){sum( (x-phi[1]\*exp(phi[2]\*t)) \* exp(phi[2]\*t))}  
dmu2 <- function(x,t,phi){sum( (x-phi[1]\*exp(phi[2]\*t)) \* phi[1]\*t\*exp(phi[2]\*t))}  
tol <- 10^{-8}  
set.seed(1)  
t <- runif(100,0,10)  
phi <- c(1,1)  
x <- 3\*exp(0.3\*t) + rnorm(100,0,7)  
plot(t,x)



for (i in 1:100)  
 {  
 dmu <-c(dmu1(x,t,phi),dmu2(x,t,phi))  
 ddmu <- matrix(c(ddmu11(t,phi),ddmu12(t,phi),ddmu21(t,phi),ddmu22(t,phi)),2,2)  
 newphi <- phi + solve(ddmu)%\*%dmu  
 err <- max(abs(newphi-phi))  
 print(c(i,phi))  
 if (err<tol) break  
 phi<-newphi  
 }

## [1] 1 1 1  
## [1] 2.00000000 0.04798841 0.99538910  
## [1] 3.00000000 0.04980944 0.89570852  
## [1] 4.0000000 0.1099059 0.6865561  
## [1] 5.0000000 0.5143744 0.2673006  
## [1] 6.0000000 2.5716430 0.6001233  
## [1] 7.0000000 0.9039368 0.5702227  
## [1] 8.0000000 1.0817402 0.4711057  
## [1] 9.0000000 1.8118112 0.3528836  
## [1] 10.0000000 2.6320627 0.3083142  
## [1] 11.000000 2.717935 0.313716  
## [1] 12.0000000 2.7193316 0.3133709  
## [1] 13.0000000 2.7193521 0.3133697  
## [1] 14.0000000 2.7193522 0.3133697

sigma <- mean((x-phi[1]\*exp(phi[2]\*t))^2)  
c(phi,sigma)

## [1] 2.7193522 0.3133697 42.5948903

ddmu<-matrix(c(ddmu11(t,phi),ddmu12(t,phi),ddmu21(t,phi),ddmu22(t,phi)),2,2)  
sigma\*solve(ddmu) # covariance matrix of hatphi

## [,1] [,2]  
## [1,] 0.171054271 -0.0074948676  
## [2,] -0.007494868 0.0003393297

2\*sigma^2/length(x) # variance of sigma^2

## [1] 36.28649

## 7. Generalized linear model (일반화선형모형)

선형회귀모형은 추정과 해석이 매우 간단하고 직관적이어서 매우 널리 사용되어 왔다. 하지만, 어떤 경우에는 선형회귀모형이 한계를 가질 때가 있다. 그것은

1. 반응변수의 범위가 제한되어 있을 때 (예) 사망여부, 수상횟수)
2. 반응변수의 분산이 평균에 의존하여 변할 때

등이다. 일반화선형모형(GLM)은 이러한 한계를 극복 혹은 보완할 수 있는 모형이다.

#### 7.1 Exponential family (지수족)

통계분석을 위해 유용한 많은 분포가 지수족이라고 하는 분포족으로 일반화할 수 있음을 알고 있을 것이다. 확률밀도(질량)함수가 다음과 같은 형태를 가질 때 지수족에 속한다 한다.

예1. -> 지수분포

예2. -> Poisson 분포

지수족의 경우 로그우도함수에 대하여 gradient vector(score function)와 expected information matrix는 각각 다음과 같이 계산된다.

단, 여기서 , 이다.

cf1) 가 선형함수면, observed information과 expected information이 일치한다. 즉, Newton's method와 Scoring method가 일치한다.

cf2) 을 독립표본이라 할 때, score function은

* Binomial model

이는 and 인 지수족이다. 또한,

그러므로,

* Poisson model

이는 and 인 지수족이다. 또한,

그러므로,

* Multinomial model

따라서,

cf) multinomial 분포의 공분산행렬은 역행렬이 존재하지 않는다. 다만, 일반화역행렬(generalized inverse)를 이용하여 계산하면, 위와 같은 결과를 얻을 수 있다. (참고문헌 Table 11.1 (개정판 Table 14.1))

* Example : Allele frequency estimation

혈액형을 결정하는 유전자는 A, B, O 세 가지 타입으로 알려져 있으며, 이들의 조합에 의해 실제로는 A(AA/AO), B(BB,BO), O(OO), AB(AB) 네 가지의 형질로 나타난다. 만약 A,B 유전자의 비율을 라 하면, 각 형질이 나타날 확률은

(1)A :

(2)B :

(3)O :

(4)AB :

가 될 것이다. 즉, 를 명의 사람에 대하여 각각의 형질이 나타난 것을 count한 것, 를 위 (1)~(4)에 나타난 각 형질의 발생확률이라 하면, 라 할 수 있다.

실제 십이지장 궤양 환자 521명을 조사한 결과, 각 혈액형이 A:186, B:38, O:284, AB:13 으로 관측되었다고 할 때, 의 MLE를 계산하여 보자.

theta<-c(0.3,0.2)  
p1<-function(theta){theta[1]\*(2-theta[1]-2\*theta[2])}  
p2<-function(theta){theta[2]\*(2-theta[2]-2\*theta[1])}  
p3<-function(theta){(1-theta[1]-theta[2])^2}  
p4<-function(theta){2\*theta[1]\*theta[2]}  
dp1<-function(theta){ c(2\*(1-theta[1]-theta[2]),-2\*theta[1]) }  
dp2<-function(theta){ c(-2\*theta[2],2\*(1-theta[1]-theta[2])) }  
dp3<-function(theta){ c(-2\*(1-theta[1]-theta[2]),-2\*(1-theta[1]-theta[2])) }  
dp4<-function(theta){ c(2\*theta[2],2\*theta[1]) }  
dl <- function(t,theta){ cbind(dp1(theta),dp2(theta),dp3(theta),dp4(theta))%\*%(t/c(p1(theta),p2(theta),p3(theta),p4(theta))) }  
I <- function(t,theta){ cbind(dp1(theta),dp2(theta),dp3(theta),dp4(theta))%\*%diag(sum(t)/c(p1(theta),p2(theta),p3(theta),p4(theta)))%\*%t(cbind(dp1(theta),dp2(theta),dp3(theta),dp4(theta))) }  
tol <- 10^{-8}  
set.seed(1)  
t <- c(186,38,284,13)  
for (i in 1:100)  
 {  
 newtheta <- theta + solve(I(t,theta))%\*%dl(t,theta)  
 err <- max(abs(newtheta-theta))  
 print(c(i,theta))  
 if (err<tol) break  
 theta<-newtheta  
 }

## [1] 1.0 0.3 0.2  
## [1] 2.0000000 0.2025672 0.0321977  
## [1] 3.00000000 0.21350164 0.04999542  
## [1] 4.0000000 0.2135911 0.0501455  
## [1] 5.00000000 0.21359094 0.05014533

c(theta,1-sum(theta))

## [1] 0.21359094 0.05014533 0.73626373

ACOV=solve(I(t,theta)) # covariance matrix of hattheta  
sqrt(c(ACOV[1,1],ACOV[2,2],ACOV[1,1]+ACOV[2,2]+2\*ACOV[1,2])) # asymptotic standard deviation of estimates

## [1] 0.013525296 0.006849005 0.014469224

#### 7.2 GLM

이 지수족에 포함된 분포로부터의 임의표본이고, 일 때 의 평균 라 하자. 단조함수 에 의하여 평균과 또다른 변수의 선형결합을 다음과 같이 모형화한다면,

score function과 expected information은 다음과 같이 주어진다.

여기서, 의 역함수 를 연결함수(link function)이라 한다. 즉, GLM은

와 같이 평균의 적절한 변환을 선형으로 모형화하는 것이다. 만약, 가 항등함수인 경우 이 모형은 우리가 잘 알고 있는 선형회귀모형이 된다.

연결함수는 평균이 가지게 되는 범위를 적절하게 조정해 주는 역할을 주로 한다. 연결함수를 어떻게 지정하느냐에 따라 의 추정량의 성질이 달라질 수 있으며, 특히 수치적으로 해를 구하는 알고리즘의 성질에도 영향을 미치게 된다.

이면, 을 정준연결함수(canonical link function)이라고 한다. 이 경우 Newton's method와 Scoring method가 일치하게 된다. (Why?)

* Logistic regression

일 때, 설명변수 에 대하여 다음과 같이 모형화하면,

, ,

cf) 연결함수 으로 하는 경우 Probit 모형이라 부른다. 또한 으로 하기도 한다. 정준연결함수는 Logit 함수이다.

* Poisson regression

일 때, 설명변수 에 대하여 다음과 같이 모형화하면,

, ,

cf) 정준연결함수는 Log 함수이다.

#### 7.3 Example

AIDS data from Australia during 1983-1986

z <- 1:14  
int <- rep(1,length(z))  
x <- c(0,1,2,3,1,4,9,18,23,31,20,25,37,45)  
plot(z,x)  
# abline(lm(x~z)$coefficients,col="blue")  
theta<-c(0,1)  
dl <- function(x,z,theta){ t(cbind(int,z))%\*%(x-exp( cbind(int,z)%\*%theta) ) }  
I <- function(x,z,theta){ t(cbind(int,z))%\*%diag(as.vector(exp(cbind(int,z)%\*%theta)))%\*%cbind(int,z) }  
tol <- 10^{-8}  
for (i in 1:100)  
 {  
 newtheta <- theta + solve(I(x,z,theta))%\*%dl(x,z,theta)  
 err <- max(abs(newtheta-theta))  
 print(c(i,theta))  
 if (err<tol) break  
 theta<-newtheta  
 }

## [1] 1 0 1  
## [1] 2.0000000 -0.9957892 0.9996948  
## [1] 3.0000000 -1.9843533 0.9988658  
## [1] 4.0000000 -2.9533422 0.9966178  
## [1] 5.0000000 -3.8695973 0.9905473  
## [1] 6.0000000 -4.6459773 0.9743377  
## [1] 7.0000000 -5.0667633 0.9323555  
## [1] 8.0000000 -4.6839986 0.8321675  
## [1] 9.0000000 -3.0139377 0.6391659  
## [1] 10.0000000 -0.8657718 0.4160759  
## [1] 11.00000000 0.05815496 0.30046811  
## [1] 12.0000000 0.3091487 0.2617294  
## [1] 13.0000000 0.3391299 0.2566130  
## [1] 14.0000000 0.3396338 0.2565236  
## [1] 15.0000000 0.3396339 0.2565236

gfit=glm(x ~ z, family = poisson()) # equivalent fitting by pre-defined function in R  
summary(gfit)

##   
## Call:  
## glm(formula = x ~ z, family = poisson())  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.21008 -1.02032 -0.69704 0.04028 2.70758   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.33963 0.25119 1.352 0.176   
## z 0.25652 0.02204 11.639 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for poisson family taken to be 1)  
##   
## Null deviance: 207.272 on 13 degrees of freedom  
## Residual deviance: 29.654 on 12 degrees of freedom  
## AIC: 86.581  
##   
## Number of Fisher Scoring iterations: 5

lines(z,exp(theta[1]+theta[2]\*z),col="red")

