full-simulation-binary

Objective

Understand maximum **simulated** likelihood with help of a simple example (binary logit model).

Introduction

The probability that the agent chooses outcome y can be expressed as an expectation (integral over the random part) (Train 2009):

$$
\begin{aligned}
P(y|x) &= Prob(I[h(x,\varepsilon) = y] = 1) && (1)\\
&= \int I[h(x,\varepsilon) = y] f(\varepsilon) d\varepsilon && (2)
\end{aligned}
$$

In random utility theory we usually assume that the observed part of utility is $\beta' x$ (a linear combination of observable factors) and that the outcome is chosen if the utility is positive (in the binary case). Therefore we can rewrite:

$$
P(y|x) = \int I[\beta' x + \varepsilon > 0] f(\varepsilon) d\varepsilon
$$

If we assume that ε is distributed logistically, such that the density is $f(\varepsilon) = e^{-\varepsilon}/(1+e^{-\varepsilon})^2$ then we can derive a closed-form solution for the integral:

$$
\begin{aligned}
P(y|x) &= \int I[\beta' x + \varepsilon > 0] f(\varepsilon) d\varepsilon && (3)\\
&= \int I[\varepsilon > -\beta' x] f(\varepsilon) d\varepsilon && (4)\\
&= \int_{\varepsilon=-\beta' x}^{\infty} f(\varepsilon) d\varepsilon && (5)\\
&= 1 - F(-\beta' x) = 1 - \frac{1}{1+e^{\beta' x}} && (6)\\
&= \frac{e^{\beta' x}}{1+e^{\beta' x}} && (7)
\end{aligned}
$$

Estimation

The goal is to learn about maximum **simulated** likelihood. This is useful for models which do not have a closed-form solution of the integral and therefore require simulation techniques to approximate it.

In our case we want to simulate $P(y|x) = \int I[\beta' x + \varepsilon > 0] f(\varepsilon) d\varepsilon$. The general strategy to achieve this is:

1. Draw from $f(\varepsilon)$ and label the rth draw ε^r
2. Compute $\beta' x + \varepsilon^r$ and evaluate the indicator function (i.e. 1 if $\beta' x + \varepsilon^r > 0$ and 0 otherwise).
3. Repeat step 1. and 2. R times.
4. The integral (and thus our probability) is simply the average $1/R \sum_{r=1}^{R} t(x, \varepsilon^r)$.

Let's implement this in R.

Implementation in R

We first simulate some data for illustration

```
simulate_binary <- function(beta = c(ASC = 0, beta1 = 1, beta2 = 2), n = 1000, seed = 0) {
  set.seed(seed)
  simulated_data <- list()

  simulated_data$beta <- beta

  x1 <- rnorm(n) # does not need to be normal!
  x2 <- rnorm(n)

  latent <- beta["ASC"] + beta["beta1"] * x1 + beta["beta2"] * x2
  choice <- as.numeric(latent + rlogis(n) > 0)

  simulated_data$data <- data.frame(x1, x2, choice)

  return(simulated_data)
}

sim_dat <- simulate_binary(n = 10e3)
beta <- sim_dat$beta
dat <- sim_dat$data

beta
#>   ASC beta1 beta2
#>     0     1     2
head(dat)
#>           x1          x2 choice
#> 1  1.2629543 -1.21955126      0
#> 2 -0.3262334 -1.20146018      0
#> 3  1.3297993 -0.49604255      1
#> 4  1.2724293  0.06693112      1
#> 5  0.4146414 -0.05694914      0
#> 6 -1.5399500  0.25558017      1
```

Using stats to illustrate

Let's use R's `stats::glm` function to test

```
fit <- glm(choice ~ x1 + x2, data = dat, family = binomial(link = "logit"))
summary(fit)
#>
#> Call:
```

```
#> glm(formula = choice ~ x1 + x2, family = binomial(link = "logit"),
#>     data = dat)
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  0.04847    0.02677   1.811   0.0702 .
#> x1           0.99444    0.03133  31.741   <2e-16 ***
#> x2           2.05691    0.04280  48.063   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 13859.8  on 9999  degrees of freedom
#> Residual deviance:  8601.2  on 9997  degrees of freedom
#> AIC: 8607.2
#>
#> Number of Fisher Scoring iterations: 5
```

Compare the estimated coefficients with our assumed beta vector. Surprise!

Closed-form own implementation

We use the `maxLik` package to impelemnt the closed-form solution. Compare the `loglik` function to the formulas above (REF). It translates almost verbatim to R code

```r
loglik <- function(param) {
  latent <- param["ASC"] + param["beta1"] * dat$x1 + param["beta2"] * dat$x2
  exp <- exp(latent)
  P_1n <- exp / (1 + exp)
  P_0n <- 1 - P_1n

  y_1n <- dat$choice
  y_0n <- 1 - y_1n

  ll <- sum(y_1n * log(P_1n) + y_0n * log(P_0n))
  cat(ll, "\n")
  ll
}

p <- function(v) {
  v <- setNames(v, c("ASC", "beta1", "beta2"))
  v
}

param <- p(c(1, 1, 1))
m <- maxLik::maxLik(loglik, start = param, method = "BFGS")
#> -5603.844
#> -5603.844
#> -5603.843
#> -5603.844
#> -5603.843
```

```
#> -5603.844
#> -5603.844
#> -5603.843
#> -5603.844
#> -5603.844
#> -5603.843
#> -5603.844
#> -5603.843
#> -5603.844
#> -5603.844
#> -5603.843
#> NaN
#> NaN
#> NaN
#> -40908.44
#> -6436.235
#> -4602.662
#> -4602.662
#> -4602.661
#> -4602.662
#> -4602.662
#> -4602.662
#> -4602.662
#> -4602.661
#> NaN
#> NaN
#> -9425.934
#> -4864.053
#> -4603.321
#> -4600.471
#> -4600.471
#> -4600.47
#> -4600.471
#> -4600.471
#> -4600.471
#> -4600.471
#> -4600.471
#> NaN
#> -8768.972
#> -4908.343
#> -4608.777
#> -4599.384
#> -4599.384
#> -4599.384
#> -4599.385
#> -4599.384
#> -4599.384
#> -4599.385
#> -4599.384
#> -4360.476
#> -4360.476
#> -4360.476
#> -4360.476
```

```
#> -4360.476
#> -4360.476
#> -4360.476
#> -4360.476
#> -4301.923
#> -4301.923
#> -4301.923
#> -4301.923
#> -4301.923
#> -4301.923
#> -4301.923
#> -4301.923
#> -4300.645
#> -4300.645
#> -4300.645
#> -4300.645
#> -4300.645
#> -4300.645
#> -4300.645
#> -4300.645
#> -4300.611
#> -4300.611
#> -4300.611
#> -4300.611
#> -4300.611
#> -4300.611
#> -4300.611
#> -4300.611
#> -5976.509
#> -4357.721
#> -4302.736
#> -4300.68
#> -4300.611
#> -4300.611
#> -4300.611
#> -4300.611
#> -4300.611
#> -4300.611
#> -4300.611
#> -4300.611
#> -4408.507
#> -4304.331
#> -4300.746
#> -4300.614
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.645
```

```
#> -4300.612
#> -4300.61
#> -5502.769
#> -4352.889
#> -4302.671
#> -4300.681
#> -4300.611
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4864.041
#> -4324.792
#> -4301.567
#> -4300.644
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4304.17
#> -4300.754
#> -4300.615
#> -4300.61
#> -4300.61
#> -4466.75
#> -4307.509
#> -4300.88
#> -4300.619
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4306.476
#> -4300.837
#> -4300.618
#> -4300.61
#> -4300.61
#> -4361.809
#> -4303.084
```

```
#> -4300.707
#> -4300.613
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
#> -4300.61
summary(m)
#> --------------------------------------------
#> Maximum Likelihood estimation
#> BFGS maximization, 69 iterations
#> Return code 0: successful convergence
#> Log-Likelihood: -4300.61
#> 3  free parameters
#> Estimates:
#>       Estimate Std. error t value Pr(> t)
#> ASC    0.04843    0.02677    1.81  0.0704 .
#> beta1  0.99435    0.03132   31.74  <2e-16 ***
#> beta2  2.05668    0.04274   48.12  <2e-16 ***
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> --------------------------------------------
```

Maximum simulated likelihood

Now to our main objective: To explore maximum simulated likelihood. Using some matrix algebra and passing a draws matrix to the `loglik` function (instead of generating the draws inside the function and/or loop over the observations)…

```
n_draws <- 1000
n_rows <- nrow(dat)
draws_matrix <- matrix(rlogis(n_draws * n_rows), nrow = n_rows, ncol = n_draws)

loglik <- function(param, epsilon = 1e-10) {
  # 10000 x 1
  latent <- param["ASC"] + param["beta1"] * dat$x1 + param["beta2"] * dat$x2

  # 10000 x n_draws
  indicator_ <- latent + draws_matrix

  indicator <- (indicator_ > 0)
  P_1n <- apply(indicator, 1, mean)
  P_1n <- pmax(pmin(P_1n, 1 - epsilon), epsilon)
  P_0n <- 1 - P_1n

  y_1n <- dat$choice
  y_0n <- 1 - y_1n

  ll <- sum(y_1n * log(P_1n) + y_0n * log(P_0n))
  cat(ll, "\n")
  ll
}
```

Clearly, our `loglik` function makes sense: Evaluated at the true parameters, we get the highest likelihood (which almost matches the non-simulated likelihood from the chapter before):

```
loglik(p(c(0, 0, 0)))
#> -6942.008
#> [1] -6942.008
loglik(p(c(1, 1, 1)))
#> -5610.319
#> [1] -5610.319
loglik(p(c(0, 1, 2)))
#> -4310.408
#> [1] -4310.408
```

Remark: The line `P_1n <- pmax(pmin(P_1n, 1 - epsilon), epsilon)` is needed because the simulated integral (depending on the parameter values passed to it) can yield $P_1n = 0$ which results in `-Inf` when taking the log. Similarly, $P_1n = 1$ would yield $P_0n = 0$ and thus the same problem. Therefore we add (or subtract) a very small value `epsilon` to 0 (from 1).

Trying to estimate (again using `maxLik`) does not seem to converge. Why?

```r
m <- maxLik::maxLik(loglik, start = param, method = "BFGS")
summary(m)
```

References

Train, Kenneth E. 2009. *Discrete Choice Methods with Simulation*. Cambridge university press.