notes

There are two main libraries for multinomial models in R:

- foreign
- nnet

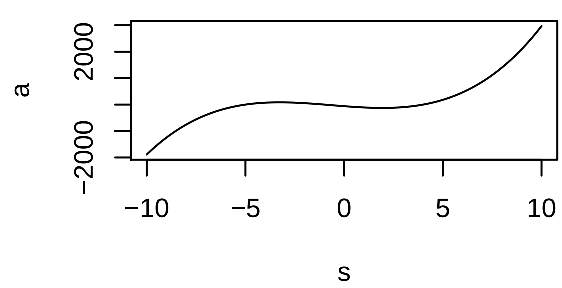
There are other mechanisms for tuning a multinomial model, but nnets are quite nice.

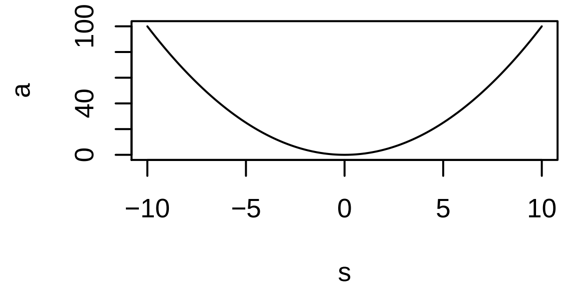
```
sub <- mydata[]
temp <- matrix(rep(0, 28*28*10), 28, 28)

for (i in 1:10){
  temp[i,] <- matrix(as.numeric(mydata[i, 2:ncol(mydata)]), 28, 28)
}
```

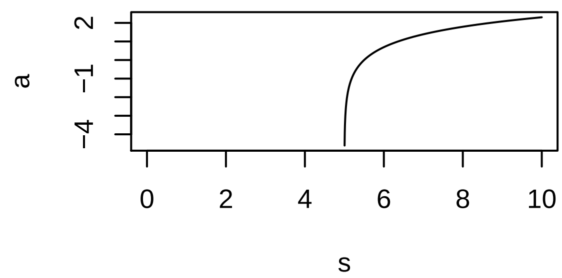
```
par(mfrow = c(2, 2)) # set up graphs in a 2x2 matrix
s <- seq(-10, 10, by = 0.01)

fx <- function(x) {3 * (x+1) * (x-4) * (x+5)}
a <- fx(s)
plot(a ~ s, type='l')

fy <- function(y) {y ^ 2}
a <- fy(s)
plot(a ~ s, type='l')

fz <- function(z) {log(z)}
a <- fz(s)
```

```
## Warning in log(z): NaNs produced
```

```
s <- seq(0, 10, by = 0.005)
plot(a ~ s, type='l')
```





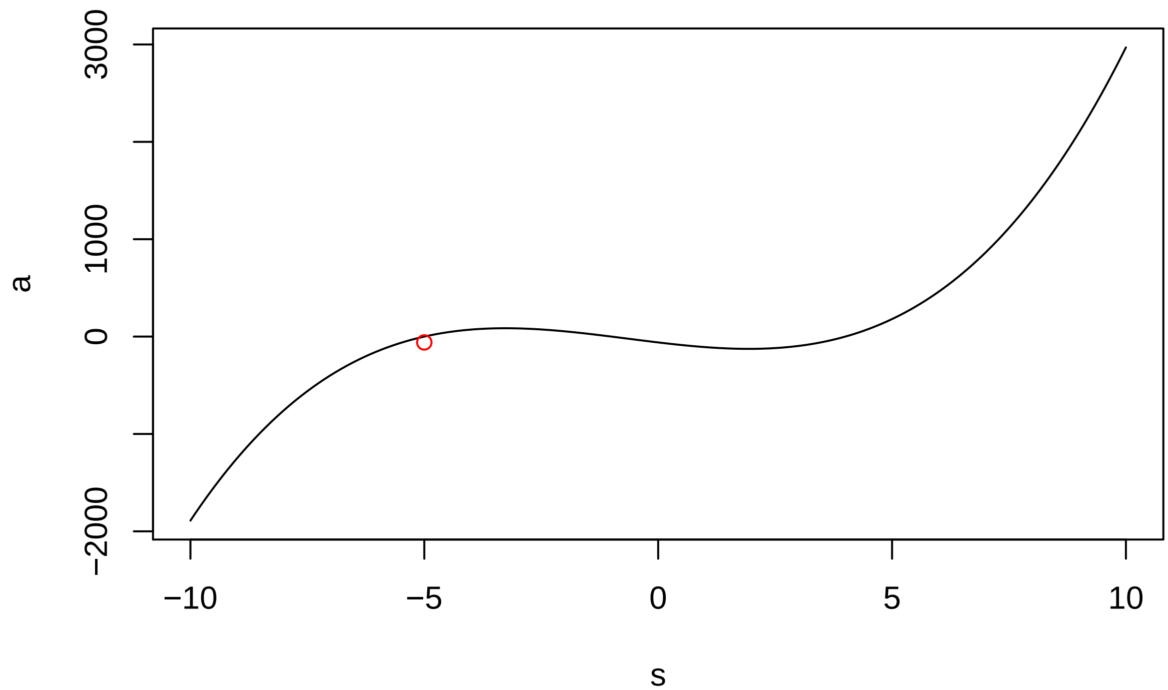


Does not pass the line test, isn't a function:

```
fqplus <- function(qplus) {sqrt(qplus)}
fqneg <- function(qneg) {-sqrt(qneg)}
s <- seq(0, 10, by=0.01)
a <- fqplus(s)
b <- fqneg(s)
s2 <- c(s,s)
plot(figukrjebcfefkctkunjrftculn
     n ~ s2, type='p')
```

Evaluating limits numerically in R

```
library(rootSolve)
fx <- function(x) {3 * (x+1) * (x-4) * (x+5)}
s <- seq(-10, 10, by=0.01)
a <- fx(s)
plot(a ~ s, type='l')

yint <- fx(0)
xint <- uniroot(fx, c(-100, 100))
points(xint$root, yint, col='red', pch=21)
```




```
xint$root
```

```
## [1] -5
```

```
yint
```

```
## [1] -60
```

Derivatives

We're going to just take super small steps to see what the change is. This is similar to Archimedes process of exhaustion... Change in y over change in x, nuff said.

$$\frac{dy}{dx} = \frac{d}{dx}(f(x)) = \frac{df}{dx}$$

We can use library(Deriv):

```
# symbolic derivative of loc(x)
library(Deriv)

myf1 <- function(x)sin(x)
Deriv(myf1)
```

```
## function (x)
## cos(x)
```

```
# symbolic derivative of 1/x
myf2 <- function(x)1/x
ans <- Deriv(myf2)

ans
```

```
## function (x)
## -(1/x^2)
```

```
ans(3) # solution evaluated at x=3
```

```
## [1] -0.1111111
```

```
# Power Rule,  px^(p-1)
myf1 <- function(x)x**n
Deriv(myf1)
```

```
## function (x)
## n * x^(n - 1)
```

```
# Addition Rule, dx(f) + dx(g)
myf2 <- function(x)3 * x + (3 * x^2 + 1)
Deriv(myf2)
```

```
## function (x)
## 3 + 6 * x
```

```
# Expponential Function, Identity
myf3 <- function(x)exp(x)

# Logarithm Function, 1/x
myf4 <- function(x)log(x)
Deriv(myf4)
```

```
## function (x)
## 1/x
```

```
# Product Rule, f'g + g'f
myf5 <- function(x)3 * x * (3 * (x^2) + 1)
Deriv(myf5)
```

```
## function (x)
## 3 * (1 + 9 * x^2)
```

```r
# Quotient Rule, f/g, (f'g - g'f)/g^2
myf6 <- function(x)3 * x / (3 * (x^2) + 1)
Deriv(myf6)
```

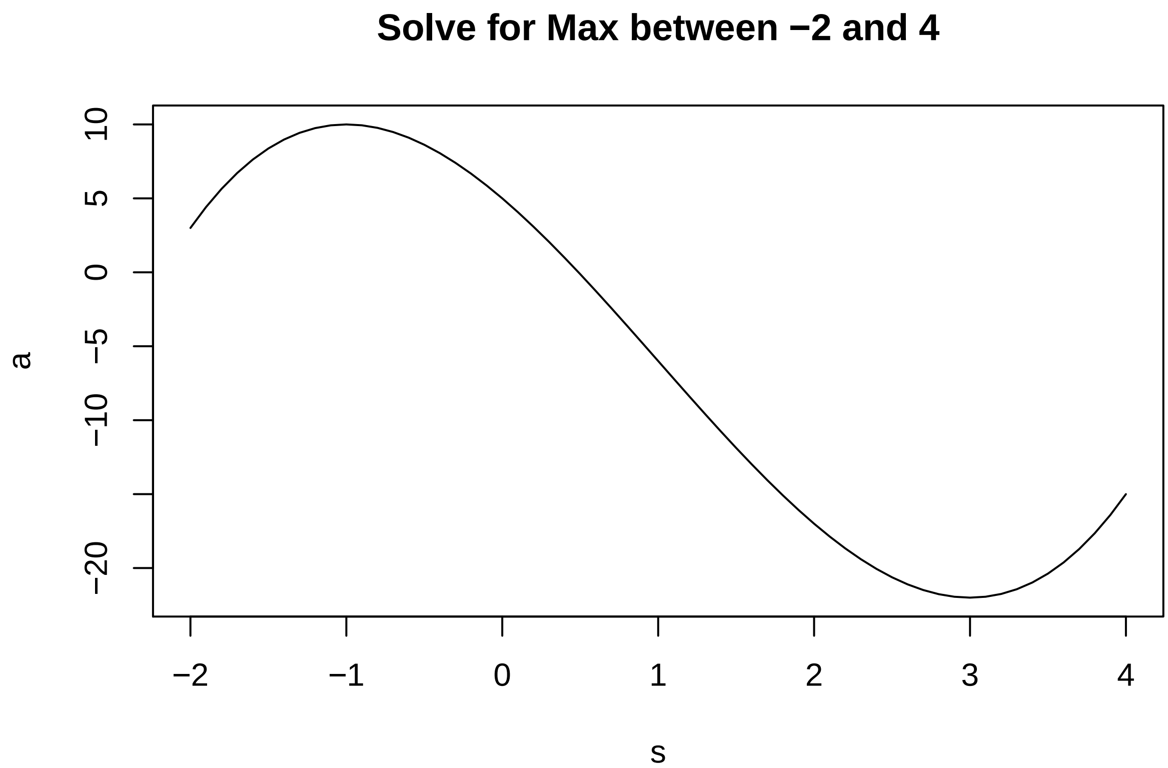
```
## function (x)
## {
##     .e1 <- x^2
##     .e2 <- 1 + 3 * .e1
##     3 * ((1 - 6 * (.e1/.e2))/.e2)
## }
```

```r
# Chain Rule, f'u * g'x
myf7 <- function(x)3 * (x^2 + 2)^2
Deriv(myf7)
```

```
## function (x)
## 12 * (x * (2 + x^2))
```

Max and Min of Functions

```r
# f(x) = x^3 - 3*x^2 - 9*x + 5   for  -2 <= x <= 4

library(mosaic)
library(mosaicCalc)
library(Deriv)
library(rootSolve)

fx <- function(x) {x^3 - 3*x^2 - 9*x + 5}
s <- seq(-2, 4, by=0.1)
a <- fx(s)

data <- cbind(s, a)
plot(data, type='l',
     main='Solve for Max between -2 and 4')
```




```
min(a)
```

```
## [1] -22
```

```
max(a)
```

```
## [1] 10
```

```
first <- Deriv(fx)
roots <- uniroot.all(first, c(-2, 4))
roots
```

```
## [1] -1.000171  3.000171
```

Integrals are anti-derivatives!

$$\int_a^b f(x)dx = \lim_{n \to \infty} \sum_{i=1}^{n} \Delta x$$

```
# R base function integrate

fx <- function(x)-2
integrate(Vectorize(fx), 1, 4) #some functions puke without vectorize
```

```
## -6 with absolute error < 6.7e-14
```

```r
fx <- function(x)exp(x^2 + x)
integrate(fx, 1, 2)
```

```
## 86.83404 with absolute error < 9.6e-13
```

Fundamental Theorem of Calculus

Part 1:

$$\int_a^b F'(x)dx = F(b) - F(a)$$

Part 2:

$$\text{If } A(x) = \int_a^x f(t)dt$$
$$A'(x) = \frac{d}{dx}\int_a^x f(t)dt = f(x)$$

```r
# Constant Multiple Rule, int(k*f) = k* int(f) + C
antiD(5 ~ x)
```

```
## function (x, C = 0)
## 5 * x + C
```

```r
# Sum Rule (separable)
antiD(x-2 + x^2 ~ x)
```

```
## function (x, C = 0)
## 1/2 * x^2 - 2 * x + 1/3 * x^3 + C
```

```r
# Power Rule, x^n -> (x^(n+1)) / (n+1)
antiD(x^3 ~ x)
```

```
## function (x, C = 0)
## 1/4 * x^4 + C
```

```r
# exp(x) => exp(x) + C
antiD(exp(x) ~ x)
```

```
## function (x, C = 0)
## exp(x) + C
```

```r
# 1/x => ln|x| + C
antiD(1/x ~ x)
```

```
## function (x, C = 0)
## 1 * log((x)) + C
```

Substitution

$$u = f(x)$$

$$x = f^{-}1(u)$$

$$du = f'(x)dx$$

$$dx = \frac{du}{f'(x)}$$

By parts

$$\int(udv) = uv - \int(vdu)$$

Two (and more) Variable Problems

Use the cubature library, it will do an adaptive integration for definite integrals.

```
library(cubature)

fxyz <- function(x) {(x[1]-2)^2 + (x[2]+3)^2 + (x[3]-4)^2 - 9}

adaptIntegrate(fxyz,
               lower = c(-10, -10, -10),
               upper = c(10, 10, 10))
```

```
## $integral
## [1] 960000
##
## $error
## [1] 1.164153e-10
##
## $functionEvaluations
## [1] 33
##
## $returnCode
## [1] 0
```