Week 12 Exercises

Aaron Palumbo

November 9, 2015

9.4.2 (Section 9.4.6)

If we wish to start out, as in Fig. 9.10, with all U and V entries set to the same value, what value minimizes the RMSE for the matrix M of our running example?

```
M = matrix(c(5, 2,  4, 4, 3,
             3, 1,  2, 4, 1,
             2, NA, 3, 1, 4,
             2, 5,  4, 3, 5,
             4, 4,  5, 4, NA),
           byrow=TRUE, ncol=5)

table(M)
```

```
## M
## 1 2 3 4 5
## 3 4 4 8 4
```

If $u_{ij} = v_{ij} = x$ then we have that the RMSE for M is:

$$
\begin{aligned}
\text{RMSE} &= & 3(1-2x^2)^2 + 4(2-2x^2)^2 + 4(3-2x^2)^2 + 8(4-2x^2)^2 + 4(5-2x^2)^2 \\
\min(\text{RMSE}) &= \tfrac{d}{dx} & 3(1-2x^2)^2 + 4(2-2x^2)^2 + 4(3-2x^2)^2 + 8(4-2x^2)^2 + 4(5-2x^2)^2 = 0 \\
0 &= \tfrac{d}{dx}\big(& 3(1-4x^2+4x^4)+ \\
& & 4(4-8x^2+4x^4)+ \\
& & 4(9-12x^2+4x^4)+ \\
& & 8(16-16x^2+4x^4)+ \\
& & 16(25-20x^2+4x^4)\big) \\
&= \tfrac{d}{dx} & 283 - 300x^2 + 92x^4 \\
&= & -600x + 368x^3 \\
&= & 8x(46x^2 - 75)
\end{aligned}
$$

From this we have: $8x = 0 \rightarrow x = 0$
and $46x^2 - 75 = 0 \rightarrow x = 1.2768848$

9.4.3 (Section 9.4.6)

Starting with the U V matrices in Fig. 9.16, do the following in order:

```
returnOpt <- function(U, V, M, uij=NA, vij=NA) {
  # uij OR vij designate the element to be optimized
  if (all(is.na(uij)) + all(is.na(vij)) != 1) {
    stop("One and only one in {uij, vij} == NA")
  }
  # Are we optimizing U or V
```

```
  if (!all(is.na(uij))) {
    r <- uij[1]; s <- uij[2]
    optU <- TRUE
  } else {
    r <- vij[1]; s <- vij[2]
    optU <- FALSE
  }

  # return proper optimizing function
  optFun <- function(optU, r, s) {
    if (optU) {
      return(
        function(U, V, M) {
          sum(V[s, ] * (M[r, ] - U[r, ] %*% V), na.rm=TRUE) /
            sum(V[s, ][!is.na(M[r, ])]^2)
        }
      )
    } else {
      return(
        function(U, V, M) {
          sum(U[ ,r] * (M[ ,s] - U %*% V[ ,s]), na.rm=TRUE) /
            sum(U[ ,r][!is.na(M[ ,s])]^2)
        }
      )
    }
  }

  f <- optFun(optU, r, s)

  if (optU) {
    U[r, s] <- 0
    U[r, s] <- f(U, V, M)
    return(U)
  } else {
    V[r, s] <- 0
    V[r, s] <- f(U, V, M)
    return(V)
  }
}
```

```
m <- 5
n <- 5
d <- 2

U <- matrix(rep(1, m*d), ncol=d)
V <- matrix(rep(1, d*n), nrow=d)

U <- returnOpt(U, V, M, uij=c(1, 1))
V <- returnOpt(U, V, M, vij=c(1, 1))
U <- returnOpt(U, V, M, uij=c(3, 1))
U
```

```
##          [,1] [,2]
```

```
## [1,] 2.600000    1
## [2,] 1.000000    1
## [3,] 1.178466    1
## [4,] 1.000000    1
## [5,] 1.000000    1
```

```
V
```

```
##        [,1] [,2] [,3] [,4] [,5]
## [1,] 1.6171    1    1    1    1
## [2,] 1.0000    1    1    1    1
```

(a) 9.4.3

Reconsider the value of u_{11}. Find it's new best value, given the changes that have been made so far.

```
U <- returnOpt(U, V, M, uij=c(1, 1))
U
```

```
##          [,1] [,2]
## [1,] 2.338378    1
## [2,] 1.000000    1
## [3,] 1.178466    1
## [4,] 1.000000    1
## [5,] 1.000000    1
```

(b) 9.4.3

Then choose the best value for u_{52}.

```
U <- returnOpt(U, V, M, uij=c(5, 2))
U
```

```
##          [,1]     [,2]
## [1,] 2.338378 1.000000
## [2,] 1.000000 1.000000
## [3,] 1.178466 1.000000
## [4,] 1.000000 1.000000
## [5,] 1.000000 3.095725
```

(c) 9.4.3

Then choose the best value for v_{22}.

```
V <- returnOpt(U, V, M, vij=c(2, 2))
V
```

```
##        [,1]     [,2] [,3] [,4] [,5]
## [1,] 1.6171 1.000000    1    1    1
## [2,] 1.0000 1.029029    1    1    1
```