개인화 추천 시스템 I 실습자료

서울대학교 통계학과[1]

2-1. 연관성 분석의 정의

연관성 분석

- 데이터 안에 존재하는 항목간의 연관성 규칙 (association rule)을 발견하는 과정
- 연관 규칙 : 상품을 구매하거나 서비스를 받는 등의 일련의 거래나 사건들의 연관성에 대한 규칙
- 연관성 분석을 마케팅에서 손님의 장바구니에 들어있는 품목간의 관계를 알아본다는 의미에서 장바구니분석 (market basket analysis)이라고도 한다.

2-2. 연관성 규칙

연관성 규칙의 예

1. 목요일 식료품 가게를 찾는 고객은 아기 기저귀와 맥주를 함께 구입하는 경향이 있다.
2. 한 회사의 전자제품을 구매하던 고객은 전자제품을 살 때 같은 회사의 제품을 사는 경향이 있다.
3. 새로 연 건축 자재점에서는 변기덮개가 많이 팔린다.

2-2. 연관성 규칙

연관성 규칙의 조건

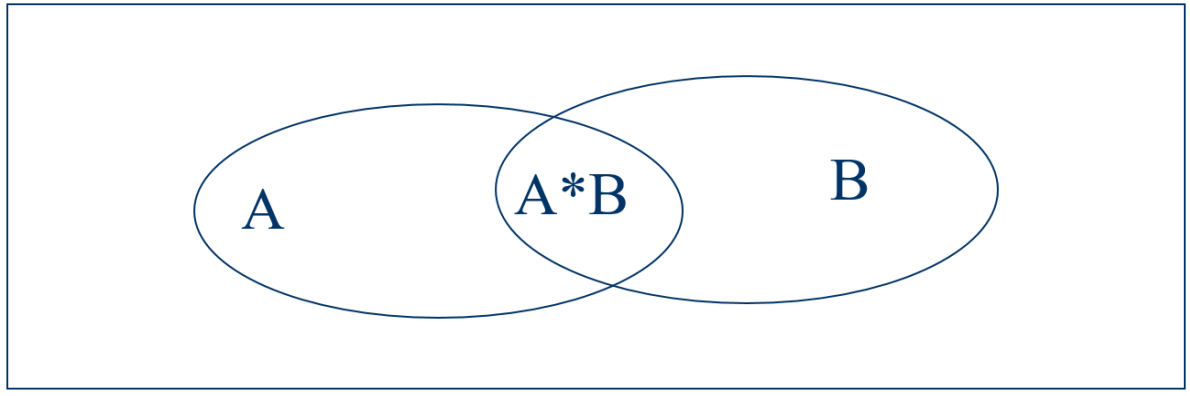
- 동시구매표로 부터 간단한 규칙 (예: 사이다를 구입하는 고객은 오렌지 쥬스를 산다)을 만들 수 있다.
- 두 품목을 함께 산 경우는 총 5번의 구매 중 2번 일어났으며 사이다를 산 3번의 구매 중 오렌지 쥬스가 2번 구매되었다.
- 연관 규칙은 "If A, then B"와 같은 형식으로 표현된다.

- 모든 "if-than" 규칙이 유용한 규칙이 아니다.
- 찾아진 규칙이 유용하게 사용되기 위하여는
  - 두 품목 (품목 A와 품목 B)이 함께 구매한 경우의 수가 일정 수준 이상 이어야 하며(일정 이상의 지지도)
  - 품목 A를 포함하는 거래 중 품목 B를 구입하는 경우의 수가 일정 수준 이상 이어야 한다.(일정 이상의 신뢰도)

2-3. 지지도, 신뢰도 및 향상도

지지도(Support)

- 두 품목 A와 B의 지지도는 전체 거래항목 중 항목 A와 항목 B가 동시에 포함하는 거래의 비율

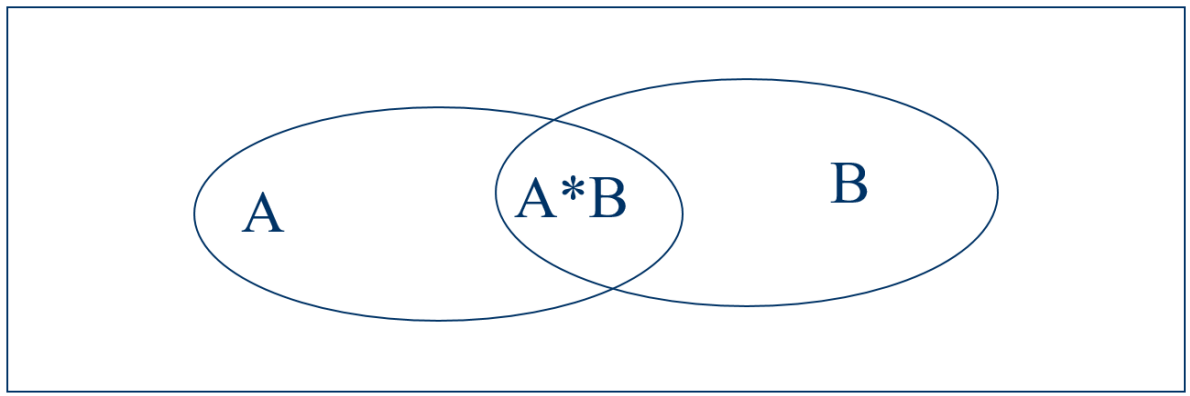


$$\text{지지도} = Pr(A \cap B) = \frac{\text{A와 B가 동시에 포함된 거래 수}}{\text{전체 거래 수}}$$

2-3. 지지도, 신뢰도 및 향상도

신뢰도(confidence)

- 연관석 규칙 "If A, then B"의 신뢰도는



$$\text{신뢰도} = \frac{Pr(A \cap B)}{P(A)} = \frac{\text{품목 A와 B를 동시에 포함하는 거래 수}}{\text{품목 A를 포함하는 거래 수}}$$

2-3. 지지도, 신뢰도 및 향상도

향상도 : 정의

- 연관성 규칙 "A이면 B이다"의 향상도는

$$\text{향상도} = \frac{\text{품목 A와 B를 포함하는 거래 수전체 거래수}}{\text{품목 A를 포함하는 거래 수} \times \text{품목 B를 포함하는 거래수}}$$
$$= \frac{P(A \cap B)}{P(A)P(B)} = \frac{P(B|A)}{P(B)} = \frac{\text{신뢰도}}{P(B)}$$

- 즉, 향상도는 품목 A가 주어지지 않았을 때의 품목 B의 확률 대비 품목 A가 주어졌을 때의 품목 B의 확률의 증가 비율 이다.
- 이 값이 클 수록, 품목 A의 구매여부가 품목 B의 구매 여부에 큰 영향을 미친다.

2-6. 연관성 분석의 실습

인터넷 쇼핑몰의 1년간 상품 구매 내역 자료(tot.csv)

- 총 37,274명의 387개 항목에 대한 구매자료이다.
- i번째 사용자가 j번째 상품을 1년 내에 구매했을 시, $tot[i, j] = 1$으로, 아닐 시에는 0.
- R workspace 파일 (ref_data_ver2) - 작업공간 불러오기를 통해 불러올 수 있다.

2-6. 연관성 분석의 실습

(1) 자료 형태 확인 및 정리 작업

```
> rownames(tot)=tot[,1]
> tot=tot[,-1]
> colnames(tot)=unq.itm
> rownames(tot)=unq.usr
> head(tot)
```

```
> head(tot)
                                 001630006001 001630005024 001630006019 001630006021 001610626004 001630028003 001630027001 001630006005
4D5441354E6A41774D446B794E413D3D            1            1            1            1            1            1            1            1
4D5441354E6A41774D54677A4E413D3D            0            0            0            0            0            0            0            0
4D5441354E6A41774D6A45784D413D3D            1            0            0            0            0            0            0            0
4D5441354E6A41774D6A67304E513D3D            0            0            0            0            0            0            0            0
4D5441354E6A41774D7A4D794E413D3D            0            0            0            0            0            0            0            0
4D5441354E6A41774E4459784E773D3D            0            0            0            0            0            0            0            0
                                 001630005019 001630028002 001630006004 001750001006 001630001006 001630002015 001610502003 001610506001
4D5441354E6A41774D446B794E413D3D            0            0            0            0            0            0            0            0
4D5441354E6A41774D54677A4E413D3D            1            1            0            0            0            0            0            0
4D5441354E6A41774D6A45784D413D3D            0            0            1            0            0            0            0            0
4D5441354E6A41774D6A67304E513D3D            0            0            0            1            1            1            1            1
4D5441354E6A41774D7A4D794E413D3D            0            0            0            0            0            0            0            0
4D5441354E6A41774E4459784E773D3D            0            0            0            0            0            0            0            0
                                 001620109008 001750001001 001610508004 001620201005 001620104016 001620104012 001620104019 001620103001
4D5441354E6A41774D446B794E413D3D            0            0            0            0            0            0            0            0
4D5441354E6A41774D54677A4E413D3D            0            0            0            0            0            0            0            0
4D5441354E6A41774D6A45784D413D3D            0            0            0            0            0            0            0            0
4D5441354E6A41774D6A67304E513D3D            0            0            0            0            0            0            0            0
4D5441354E6A41774D7A4D794E413D3D            1            1            1            1            0            0            0            0
4D5441354E6A41774E4459784E773D3D            0            0            0            0            1            1            1            1
```

구매내역 없음

구매내역 존재

2-6. 연관성 분석의 실습

(2) 모형 적합

- 구매 여부에 대해 각 상품들 간의 연관성 규칙을 찾고자 한다.
- 지지도 0.01, 신뢰도 0.5를 기준으로 찾음
- apriori(data, parameter=list(supp= , conf=), appearance=)
  - arules 패키지에 포함된 함수
  - 입력하고자 하는 자료형과 연관규칙의 대상, 지지도, 신뢰도를 설정
  - 입력 가능한 자료형 : matrix
  - parameter : 지지도, 신뢰도 결정
  - appearance : 결과 출력 형식 지정

2-6. 연관성 분석의 실습

(2) 모형 적합

```
> library(arules)
> colnames(tot)=unq.itm.name
> rules=apriori(as.matrix(tot), parameter=list(supp=0.01, conf=0.5))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE       5    0.01      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 372

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[387 item(s), 37274 transaction(s)] done [0.01s].
sorting and recoding items ... [105 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [9 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

2-6. 연관성 분석의 실습

(2) 모형 적합

- rules.sorted = sort(rules, by="lift")
  : 결과를 해석하기 좋게 향상도 순으로 정렬

```
> rules.sorted=sort(rules, by="lift")
> inspect(rules.sorted)
    lhs                                                  rhs                                                  support    confidence lift
[1] {완구/교육/보드게임  블록/퍼즐/교육/학습완구  레고시리즈[LEGO] } => {완구/교육/보드게임  블록/퍼즐/교육/학습완구  블록 } 0.01293127 0.5452489  7.163767
[2] {출산/임부/유아용품  유아동 화장품  바스/샴푸/비누 }          => {출산/임부/유아용품  유아동 화장품  로션/크림 }      0.02733809 0.5247168  2.442649
[3] {출산/임부/유아용품  수유/이유용품  젖병소독기/건조대 }       => {출산/임부/유아용품  수유/이유용품  젖병/젖꼭지 }    0.01421903 0.5391658  2.268013
[4] {출산/임부/유아용품  수유/이유용품  보온병/분유케이스 }       => {출산/임부/유아용품  수유/이유용품  젖병/젖꼭지 }    0.01250201 0.5356322  2.253149
[5] {출산/임부/유아용품  수유/이유용품  유축기}                 => {출산/임부/유아용품  수유/이유용품  젖병/젖꼭지 }    0.01687503 0.5285714  2.223448
[6] {출산/임부/유아용품  수유/이유용품  보틀워머/치발기}        => {출산/임부/유아용품  수유/이유용품  젖병/젖꼭지 }    0.01690186 0.5215232  2.193799
[7] {출산/임부/유아용품  수유/이유용품  젖병세정제/젖병솔 }      => {출산/임부/유아용품  수유/이유용품  젖병/젖꼭지 }    0.02985996 0.5131397  2.158534
[8] {기저귀/분유  분유  남양유업 }                              => {기저귀/분유  기저귀  하기스 }                    0.02194559 0.6054774  1.538416
[9] {기저귀/분유  분유  매일유업 }                              => {기저귀/분유  기저귀  하기스 }                    0.02283093 0.5525974  1.404057
```

총 9개의 연관규칙이 생성
같은 중분류의 상품들 간의 연관규칙들이 생성되었다.

2-6. 연관성 분석의 실습

(2) 모형 적합

- rules.sub=subset(rules, rhs %pin% c("크림"))
  : appriori 함수로 얻은 여러 연관성 규칙 중 관심있는 연관규칙 출력

```
> rules.sub = subset(rules, rhs %pin% c("크림"))
> inspect(rules.sub)
    lhs                                                rhs                                                 support    confidence lift
[1] {출산/임부/유아용품  유아동 화장품  바스/샴푸/비누 } => {출산/임부/유아용품  유아동 화장품  로션/크림 } 0.02733809 0.5247168  2.442649
```

앞의 규칙들 중, 오른쪽 규칙에 "크림" 이 포함된 규칙만 출력할 수 있다.

2-6. 연관성 분석의 실습

직접 해보기

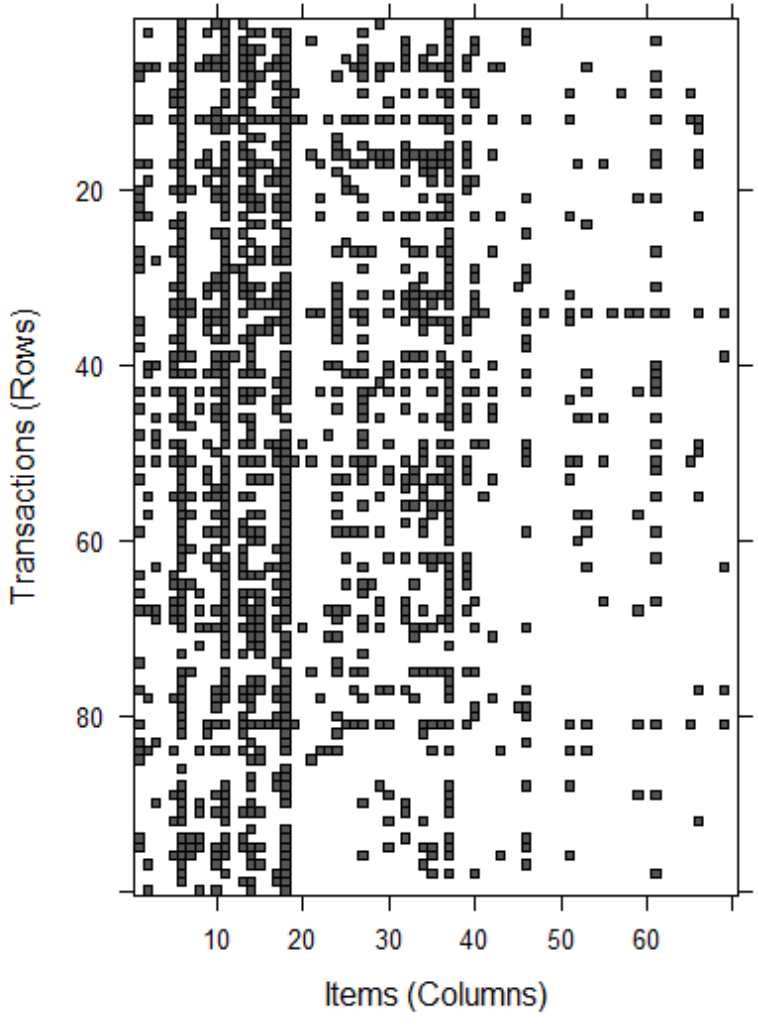
- 인터넷 쇼핑몰 자료 (shopping.csv)
  - 인터넷 쇼핑몰에서 판매되는 상품들을 70가지 품목으로 중분류
  - ex) 의류, 화장품, 전자제품, 건강제품 등
  - 인터넷 쇼핑몰 회원 중 임의 추출된 10,000명을 대상으로 2013년 한 해 동안 70가지 품목에 대한 구매여부를 저장

자료 정제 힌트 : 각 고객이 상품을 샀는지의 여부만을 이용하여 자료를 만든다.

```
> colnames(shopping)=c("customer","goods","times")
> i <- shopping$customer
> j <- shopping$goods
> shopping.m=sparseMatrix(i,j, dims = c(max(i),max(j)), x=1)
```

2-6. 연관성 분석의 실습

고객 일부(100명)에 대한 70개 상품 구매여부



Reference

[1] 박창이 · 김용대 · 김진석 · 송종우 · 최호식. **R을 이용한 데이터마이닝.** 교우사, 2011. pp.227-249.

[2] Hahsler, M., Buchta, C., Hornik, K., Johnson, I. and Borgelt, C. *Package 'arules'*, 2017.

3-1. Market Basket 자료 분석

Market Basket 자료 분석

- 고객의 historical 구매 자료에 대해 생각해보자.
- Market Basket 자료 분석은 고객의 이전 구매 이력을 기반으로 다음 시점에 어떤 상품을 구매할 것인지 예측하는 문제.

3-2. Conditional regression approach

표기법

- U : 고객 집합, I : 상품 집합
- s_{ui} : 고객 u의 이전 구매 자료로부터의 상품 i 에 대한 구매 정보. 여기서 $u \in U$, $i \in I$.
  - ex) 구매 정보는 구매 여부(0 또는 1), 구매 횟수, 구매 비율 등을 말함.
- Y_{ui} : 고객 u가 다음 시점에 상품 i를 구매하면 1, 아니면 0.

3-2. Conditional regression approach

- 고객상품에 대한 정보를 추가하면 더 좋은 예측력을 가짐.
  - X_u : 고객 u에 대한 입력변수. ex) 성별, 나이, 거주지역 등

- 로지스틱 회귀 모형

$$P(Y_{ui} = 1|s_u, X_u) = \frac{exp(\beta_0^{(i)} + \beta_1^{(i)'} s_u + \beta_2^{(i)'} X_u)}{1 + exp(\beta_0^{(i)} + \beta_1^{(i)'} s_u + +\beta_2^{(i)'} X_u)}$$

  - 변수들 간의 상호작용(interaction)을 고려하기 위해 부스팅 모형을 사용할 수 있음.

- 문제점
  - 자료가 작은 경우 분산이 커짐.
  - 상품의 갯수가 많을 때 모형 $|I|$개를 적합시켜야 함.

3-2. Conditional regression approach

- 위의 문제점을 해결하기 위해 (고객, 상품) 쌍에 대해 모형화 할 수 있음.
- 이때는 상품에 대한 정보를 추가 할 수 있음.
  - Z_i : 상품 i에 대한 입력변수. ex) 제조사, 가격, 장르, 상품카테고리, 인기도 등

- 로지스틱 회귀 모형

$$P(Y_{ui} = 1 | s_u, X_u, Z_i)$$
$$= \frac{exp(\beta_0 + \beta_1' s_u + \beta_2' X_u + \beta_3' Z_i + \mathbf{B}_1 s_u * Z_i + \mathbf{B}_2 X_u * Z_i)}{1 + exp(\beta_0 + \beta_1' s_u + \beta_2' X_u + \beta_3' Z_i + \mathbf{B}_1 s_u * Z_i + \mathbf{B}_2 X_u * Z_i)}$$

  - 마찬가지로 부스팅 모형을 사용하면 예측력이 더 좋음.

3-3. 분류 예측 평가방법

Precision vs Recall

- Precision :
  추천한 모든 상품들 중 실제 평가가 좋은 상품의 비율

$$Precision = \frac{tp}{tp + fp} = \frac{|good\ movies\ recommended|}{|all\ recommendations|}$$

- Recall :
  평가가 좋은 모든 상품들 중 추천된 상품의 비율

$$Recall = \frac{tp}{tp + fn} = \frac{|good\ movies\ recommended|}{|all\ good\ movies|}$$

3-3. 분류 예측 평가방법

F1 score

- Precision과 Recall은 서로 Trade-off 관계.
- F1 score 는 Precision과 Recall의 조화평균

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

3-5. Market Basket Regression의 실습

카드사 고객의 3개월 간 업종별 구매자료(card.csv)

- 총 60,897명 고객의 30개 업종에 대한 총 5,027,693건의 카드 사용 내역 자료.
- 2014년 5월에서 7월까지의 자료.
- 5,6월 카드 사용 내역을 가지고, 7월에 쿠폰을 발행하려고 한다. 어떤 고객에게 어떤 업종에 대한 쿠폰을 발행해야 더 많은 반응을 얻을수 있을까?
- 변수는 고객번호(CLNN), 가맹점 업종명(MCT_RY_NM), 승인일자(APV_TS_D), 성별(SEX_CCD), 연령(CLN_AGE), 연소득(천만원, AVG_Y_INA) 임.

3-5. Market Basket Regression의 실습

ex) 고객번호 "P223597622" 의 5,6월 과 7월 카드 사용 내역

- 5,6월

	CLNN	MCT_RY_NM	APV_TS_D
1	P223597622	주유소	20140506
2	P223597622	할인점/슈퍼마켓	20140507
3	P223597622	한식	20140519
4	P223597622	할인점/슈퍼마켓	20140522
5	P223597622	주유소	20140522
6	P223597622	제과점	20140522
7	P223597622	한식	20140523
8	P223597622	한식	20140527
9	P223597622	일반대중음식(까페생맥주치킨)	20140529
10	P223597622	한식	20140602
11	P223597622	일반대중음식(까페생맥주치킨)	20140602
12	P223597622	식품잡화	20140605
13	P223597622	편의점	20140605
14	P223597622	남.여기성복	20140606
15	P223597622	할인점/슈퍼마켓	20140611
16	P223597622	한식	20140612
17	P223597622	할인점/슈퍼마켓	20140617
18	P223597622	한식	20140628

- 7월

	CLNN	MCT_RY_NM	APV_TS_D
1	P223597622	한식	20140725
2	P223597622	일식	20140725
3	P223597622	할인점/슈퍼마켓	20140726
4	P223597622	일반대중음식(까페생맥주치킨)	20140727
5	P223597622	식품잡화	20140727
6	P223597622	한식	20140728

- 고객번호 "P22359622"의 정보

	CLNN	SEX_CCD	CLN_AGE	AVG_Y_INA
49113	P223597622	M	33	37

3-5. Market Basket Regression의 실습

(1) 자료 형태 확인 및 정리 작업

- 자료 형태

```
> head(data)
        CLNN MCT_RY_NM APV_TS_D SEX_CCD CLN_AGE AVG_Y_INA
1 P958107922      한식 20140616       F      35        57
2 P724103032    주유소 20140514       F      29        35
3 P012419254    주유소 20140526       M      44        40
4 P724103032    주유소 20140526       F      29        35
5 P346324532    주유소 20140607       M      25        26
6 P014022104    주유소 20140621       M      41        69
```

3-5. Market Basket Regression의 실습

(1) 자료 형태 확인 및 정리 작업

- 승인일자에서 월만 추출

```
> data1 <- data %>% mutate(month=ifelse(APV_TS_D>20140700, 7, 6),
+                          month=ifelse(APV_TS_D<20140601, 5, month)) %>% select(-APV_TS_D)
```

- 고객 정보 추출
- 고객 나이와 성별변수를 더미변수로 변환.

```
> user <- data1 %>% select(CLNN, SEX_CCD, CLN_AGE, AVG_Y_INA) %>%
+   distinct(CLNN, .keep_all=TRUE)
> user <- user %>% mutate(age2 =ifelse( (CLN_AGE>=40 & CLN_AGE <60), 1, 0),
+                         age3 =ifelse(CLN_AGE >=60, 1, 0)) %>% select(-CLN_AGE)
> user$SEX_CCD <- ifelse(user$SEX_CCD =="F", 1, 0)
>
> head(user)
        CLNN SEX_CCD AVG_Y_INA age2 age3
1 P958107922       1        57    0    0
2 P724103032       1        35    0    0
3 P012419254       0        40    1    0
4 P346324532       0        26    0    0
5 P014022104       0        69    1    0
6 P982928532       0        46    0    0
```

3-5. Market Basket Regression의 실습

(1) 자료 형태 확인 및 정리 작업

- 5,6월 자료로 설명변수를 만듦.

```
> library(tidyr)
> input <- data1 %>% filter(month !=7) %>% group_by(CLNN, MCT_RY_NM) %>%
+   summarise(count=n()) %>% spread(MCT_RY_NM, count) %>% ungroup()
> input <- input %>% inner_join(user, by="CLNN")
> input[is.na(input)]=0
```

고객 번호

고객의 5,6월 해당 상품 구매 횟수

```
> head(input)
# A tibble: 6 x 35
        CLNN CATV상품판매 LPG가스 개인병원 `결제대행(PG)` 공연장극장 남.여기성복 농수산물 문구용품
      <fctr>       <dbl>   <dbl>    <dbl>          <dbl>      <dbl>      <dbl>    <dbl>     <dbl>
1 P000024320           0       0        1              0          0          0        0         0
2 P000035010           0       0        0              0          0          0        0         0
3 P000035968           0       0        0              0          0          0        0         0
4 P000038512           0       0        0              0          0          0        0         0
5 P000045591           1       0        0              0          0          0        0         0
6 P000051783           0       0        0              0          0          1        0         0
```

...

```
  커피전문점 패스트푸드 편의점  한식 `할인점/슈퍼마켓` 화장품 SEX_CCD AVG_Y_INA  age2  age3
       <dbl>      <dbl>  <dbl> <dbl>          <dbl>  <dbl>     <dbl>  <int> <dbl> <dbl>
1          0          0      1     1              5      0         0     57     0     1
2          0          0      0     2              0      0         0     30     0     1
3          0          0      1    13              0      0         0     32     0     1
4          0          1      0     0              1      0         1     18     0     1
5          0          1      4    17              2      0         0     29     0     1
6          0          0      0     0              6      0         1     21     0     1
```

고객에 대한 성별, 수입, 나이 더미 변수

3-5. Market Basket Regression의 실습

(1) 자료 형태 확인 및 정리 작업

- 7월 자료로 종속변수들을 만듦.

```
> label <- data1 %>% filter(month==7) %>% group_by(CLNN, MCT_RY_NM) %>% summarise(label=1)%>% ungroup()
> label <- label %>% group_by(CLNN) %>% spread(MCT_RY_NM, label) %>% ungroup()
> label[is.na(label)]=0
> head(label)
# A tibble: 6 x 31
        CLNN CATV상품판매 LPG가스 개인병원 `결제대행(PG)` 공연장극장 남.여기성복 농수산물 문구용품
      <fctr>        <dbl>   <dbl>    <dbl>          <dbl>      <dbl>       <dbl>    <dbl>    <dbl>
1 P000024320            0       0        1              0          0           0        0        0
2 P000035010            0       0        0              0          0           0        0        0
3 P000035968            0       0        0              0          0           0        0        0
4 P000038512            0       0        1              0          0           0        0        0
5 P000045591            0       0        0              0          0           0        0        0
6 P000051783            0       0        0              0          0           0        0        0
```

고객이 7월에 해당상품을 샀으면 1, 아니면 0.

3-5. Market Basket Regression의 실습

(1) 자료 형태 확인 및 정리 작업

- 자료의 30%를 랜덤하게 뽑아 검증자료로 사용.

```
> set.seed(1001)
> idx.ts = sample(1:nrow(input), round(nrow(input)*0.3))
> idx.ts = sort(idx.ts)
> train=input[-idx.ts,]; label.tr = label[-idx.ts,]
> test=input[idx.ts,]; label.ts = label[idx.ts,]
```

- input 과 label matrix의 고객번호는 따로 저장.

```
> user.tr = train$CLNN; user.ts = test$CLNN
> train = train[,-1]; test = test[,-1]
> label.tr = label.tr[,-1]; label.ts = label.ts[,-1]
```

3-5. Market Basket Regression의 실습

(2) 모형 1: 구매횟수가 많은 품목 추천

- 구매횟수가 많은 품목은 순서대로 할인점/슈퍼마켓, 한식, 편의점이다.

```
> item.count=apply(train[,1:30], 2, sum)
> item.count=sort(item.count, decreasing = T)
> head(item.count)
            할인점/슈퍼마켓                       한식                     편의점
                      492053                     345368                     312618
일반대중음식(까페생맥주치킨)                 커피전문점                      백화점
                      209789                     101499                      97116
```

3-5. Market Basket Regression의 실습

(3) 모형 2: 로지스틱 회귀 모형 적합, 예측

- 30개의 항목별 구매여부를 ridge 패널티를 부여한 로지스틱 회귀 모형으로 예측하고자 한다.
- glmnet 패키지에 내장된 glmnet 함수 이용.
- 30개의 모형에 일관된 조절모수(λ) 0.02 적용.
- predict 함수를 사용하여 평가자료로 항목별 구매 확률 추정하여 저장.

```
> p.logis = label.ts
> library(glmnet)
> for(i in 1:30){
+   lm=glmnet(x=as.matrix(train), y=as.matrix(label.tr[,i]), family="binomial", alpha=0, lambda = 0.02)
+   p.logis[,i]=predict(lm, as.matrix(test), type="response")
+   rm(lm); gc()
+ }
```

family="binomial" : 로지스틱 회귀분석

3-5. Market Basket Regression의 실습

(4) 모형 3: 부스팅 모형 적합, 예측

- xgboost 패키지의 xgboost 함수 사용
- X=xgb.DMatrix(as.matrix(train), label=as.matrix(label.tr)[,i])
  - 먼저 입력변수와 출력변수를 xgb.Dmatrix 형태로 변환
- model <- xgboost(X, max_depth=3, eta=0.1, nrounds = 200, objective="binary:logistic", verbose = F)
  - 모형 저장
  - max_depth : 결합하는 tree의 깊이
  - eta : 부스팅의 step-size
  - nrounds : 트리의 갯수

3-5. Market Basket Regression의 실습

(4) 모형 3: 부스팅 모형 적합, 예측

- predict 함수로 재구매 확률 예측

```
> p.boost = label.ts
> library(xgboost)
> for(i in 1:30){
+   X=xgb.DMatrix(as.matrix(train), label=as.matrix(label.tr)[,i])
+   model <- xgboost(X, max_depth=3, eta=0.1, nrounds = 200, objective="binary:logistic", verbose = F)
+   p.boost[,i]=predict(model, as.matrix(test), type="response")
+   rm(model); gc()
+ }
```

3-5. Market Basket Regression의 실습

(5) 모형 비교

- 5,6월과 7월 거래수 및 고객수 (검증자료)

	5,6월	7월	Total
전체거래수	998,371	510,149	1,508,520
전체고객수	18,264		

- 5,6월 데이터를 이용해 7월 구매가능성 예측 수행

<시나리오>
모형1 vs 모형 2,3
1. 구매횟수가 가장 많은 품목 추천 vs 사용자별로 가장 선호도가 높은 품목 추천
2. 구매횟수가 많은 상위 2개 품목 추천 vs 사용자별로 선호도가 높은 상위 2개 품목 추천
3. 구매횟수가 많은 상위 3개 품목 추천 vs 사용자별로 선호도가 높은 상위 3개 품목 추천
4. 구매횟수가 많지 않은 특정 3개 품목 추천 vs 품목별로 구매가능성 높은 일부 고객에게 추천

3-5. Market Basket Regression의 실습

(5) 모형 비교

- 결과

		모형1				모형2				
시나리오	추천품목	추천수	추천품목수	Actual 구매	Hit Ratio	추천수	추천품목수	Hit 수	Hit Ratio	모형1대비 Lift
1	할인점/슈퍼마켓	18,264	1	15,317	83.86%	18,264	28	16,988	93.01%	10.91%
2	할인점/슈퍼마켓 한식	36,528	2	30,637	83.87%	36,528	30	32,306	88.44%	5.45%
3	할인점/슈퍼마켓 한식 편의점	54,792	3	41,777	76.25%	54,792	30	46,362	84.61%	10.97%
4	커피전문점 백화점 제과점	54,792	3	27,029	49.33%	21,000	3	14,237	67.80%	37.43%
								(Lift : Hit Ratio 향상도로 정의)		

<시나리오 4 상세자료>

			모형2		
	구매고객수		추천수	Hit 수	Hit Ratio
커피전문점	7460	40.85%	7000	5039	71.99%
백화점	6144	33.64%	7000	4097	58.53%
제과점	8429	46.15%	7000	5101	72.87%

3-5. Market Basket Regression의 실습

(5) 모형 비교

- 결과(계속)

		모형3				
시나리오	추천품목	추천수	추천품목수	Hit 수	Hit Ratio	모형2대비 Lift
1	할인점/슈퍼마켓	18,264	26	17,193	94.14%	1.21%
2	할인점/슈퍼마켓 한식	36,528	29	32,511	89.00%	0.63%
3	할인점/슈퍼마켓 한식 편의점	54,792	30	46,761	85.34%	0.86%
4	커피전문점 백화점 제과점	21,000	3	14,431	68.72%	1.36%
				(Lift : Hit Ratio 향상도로 정의)		

<시나리오 4 상세자료>

			모형3		
	구매고객수		추천수	Hit 수	Hit Ratio
커피전문점	7460	40.85%	7000	5110	73.00%
백화점	6144	33.64%	7000	4187	59.81%
제과점	8429	46.15%	7000	5134	73.34%

3-5. Market Basket Regression의 실습

(5) 모형 비교

- 코드
  - 모형 1: 구매횟수 많은 품목 추천

```
#추천횟수 많은 품목 추천
real.item=colSums(label.ts)

real.item[29]/length(user.ts) #할인점/슈퍼마켓 추천
sum(real.item[c(29,28)])/(2*length(user.ts)) #할인점/슈퍼마켓, 한식 추천
sum(real.item[c(29,28,27)])/(3*length(user.ts)) #할인점/슈퍼마켓,한식, 편의점 추천
sum(real.item[c(25,27,21)])/(3*length(user.ts)) #커피전문점, 백화점, 제과점
```

3-5. Market Basket Regression의 실습

(5) 모형 비교

- 코드
  - 모형 2: 사용자별 가장 선호도 높은 품목 추천

```
#user별 첫번째, 두번째, 세번째 확률 높은 아이템 인덱스 추출
index1=apply(p.logis, 1, function(x) sort.int(t(x), index.return=TRUE, decreasing = T)$ix[1])
index2=apply(p.logis, 1, function(x) sort.int(t(x), index.return=TRUE, decreasing = T)$ix[2])
index3=apply(p.logis, 1, function(x) sort.int(t(x), index.return=TRUE, decreasing = T)$ix[3])

#Hit ratio (Precision)
sum(as.matrix(label.ts)[cbind(1:nrow(label.ts),index1)])/length(user.ts)
(sum(as.matrix(label.ts)[cbind(1:nrow(label.ts),index1)]) + sum(as.matrix(label.ts)[cbind(1:nrow(label.ts),index2)]))/
  (2*length(user.ts))
(sum(as.matrix(label.ts)[cbind(1:nrow(label.ts),index1)]) + sum(as.matrix(label.ts)[cbind(1:nrow(label.ts),index2)])+
    sum(as.matrix(label.ts)[cbind(1:nrow(label.ts),index3)]))/
  (3*length(user.ts))

#추천 품목수
length(unique(index1))
length(unique(index2))
length(unique(index3))


#품목별로 구매가능성 높은 일부 고객에게 추천
#커피전문점, 백화점, 제과점
(sum(label.ts[sort.int(t(p.logis[,25]), index.return=TRUE, decreasing = T)$ix[1:7000],25]) +
    sum(label.ts[sort.int(t(p.logis[,9]), index.return=TRUE, decreasing = T)$ix[1:7000],9]) +
    sum(label.ts[sort.int(t(p.logis[,21]), index.return=TRUE, decreasing = T)$ix[1:7000],21])) / (7000*3)
```

3-5. Market Basket Regression의 실습

직접 해보기(온라인 식료품 상점 구매 자료(instacart.csv))

- 온라인 식료품 상점 구매 자료의 일부 고객에 대한 자료.
- **각 구매시점 별 동시구매 정보가 있는 것이 특징**
- 고객들 장바구니의 상품 구매 자료에서 (고객, 상품) 정보를 추출한 것이다.
- 고객 평균 장바구니의 갯수는 16개, 한 장바구니 당 평균 상품 갯수는 10개.
- (고객, 상품) 쌍의 자료로 849,576쌍이고, 고객이 13,120명, 상품이 37,401개.
- 입력변수는 총 34개 (뒷장 참조), 출력변수는 고객이 마지막 주문에 상품을 재구매 하면 1, 아니면 0.

3-5. Market Basket Regression의 실습

- 고객에 대한 입력변수 (X_u)

순번	변수명	설명
1	user_orders	고객의 총 장바구니 주문수
2	user_total_products	고객의 주문한 총 상품수
3	user_distinct_products	고객의 unique한 구매 상품 수
4	user_average_basket	고객 장바구니의 상품 수 평균
5	user_reorder_ratio	두번째 주문부터의 총 주문상품수 대비 재구매상품 비율
6	user_period	고객의 첫 주문부터 직전 주문이 일어난 때까지의 기간
7	user_mean_days_since_prior	고객의 주문이 이전주문으로부터 몇일 걸리는지 평균
8	user_last2	고객 최근 2번째 주문 후 지난 일수
9	user_last3	고객 최근 3번째 주문 후 지난 일수
10	user_mean_order_dow	고객 주문 요일 평균
11	user_mean_order_hour_of_day	고객 주문 시간 평균
12	order_number	고객의 예측하고하 하는 주문 번호
13	order_dow	고객의 예측하고자 하는 주문 요일
14	order_hour_of_day	고객의 예측하고자 하는 주문 시간
15	days_since_prior_order	고객의 최근 주문 후 지난 일수

3-5. Market Basket Regression의 실습

- 상품에 대한 입력변수 (Z_i)

순번	변수명	설명
1	prod_orders	상품의 총 주문수
2	prod_reorders	상품 총 재구매 수
3	prod_reorder_ratio	상품 구매수 대비 재 구매수
4	prod_reorder_probability	상품 처음 주문 상품 수 대비 두번째 주문 상품 수 비율
5	prod_reorder_times	상품 처음 주문 상품 수 대비 총 재주문 상품 수
6	prod_add_to_cart_mean	상품이 몇번째로 장바구니에 담기는지 평균
7	prod_users_unq	상품 구매한 unique 고객수
8	prod_users_ unq_reordered	상품 재구매한 unique 고객수

3-5. Market Basket Regression의 실습

- (고객, 상품)에 대한 입력변수 (Y_{ui})

순번	변수명	설명
1	up_orders	(고객, 상품)의 총 주문수
2	up_reordered_sum	(고객, 상품)의 재구매 횟수
3	up_first_order	(고객, 상품)의 첫 주문의 주문 넘버
4	up_last_order	(고객, 상품)의 최근 주문의 주문 넘버
5	up_average_cart_position	(고객, 상품) 장바구니에 몇번째로 넣는지 평균
6	up_order_rate	고객 주문수 대비 (고객, 상품)의 총 주문수
7	up_orders_since_last_order	예측하고자 하는 주문이 (고객, 상품) 최근 주문 후 몇번째 주문인지
8	up_order_rate_since_first_order	(고객, 상품) 첫 주문후 주문비율
9	up_days	최근 고객이 상품 구매 후 지난 일수
10	up_order_dow_mean	(고객, 상품) 구매 요일 평균
11	up_days_since_prior_order_mean	(고객, 상품) 주문의 days since prior order 평균

3-5. Market Basket Regression의 실습

고객, 상품 번호

고객에 대한 입력변수

	user_id	product_id	user_orders	user_total_products	user_distinct_products	user_average_basket
1	9	311	3	76	58	25.33333
2	9	481	3	76	58	25.33333
3	9	1559	3	76	58	25.33333
4	9	2732	3	76	58	25.33333
5	9	3634	3	76	58	25.33333
6	9	4957	3	76	58	25.33333

	user_reorder_ratio	user_period	user_mean_days_since_prior	user_last2	user_last3	user_mean_order_dow
1	0.3913043	36	18	60	66	2
2	0.3913043	36	18	60	66	2
3	0.3913043	36	18	60	66	2
4	0.3913043	36	18	60	66	2
5	0.3913043	36	18	60	66	2
6	0.3913043	36	18	60	66	2

⋮

(상품에 대한 입력변수)

	up_order_rate_since_first_order	up_days	up_order_dow_mean	up_days_since_prior_order_mean	reordered
1	0.3333333	66	1.0	31	0
2	0.6666667	30	3.0	30	0
3	1.0000000	30	5.0	30	1
4	1.0000000	30	5.0	30	0
5	1.0000000	30	5.0	30	0
6	0.6666667	60	0.5	6	0

(고객, 상품)에 대한 입력변수

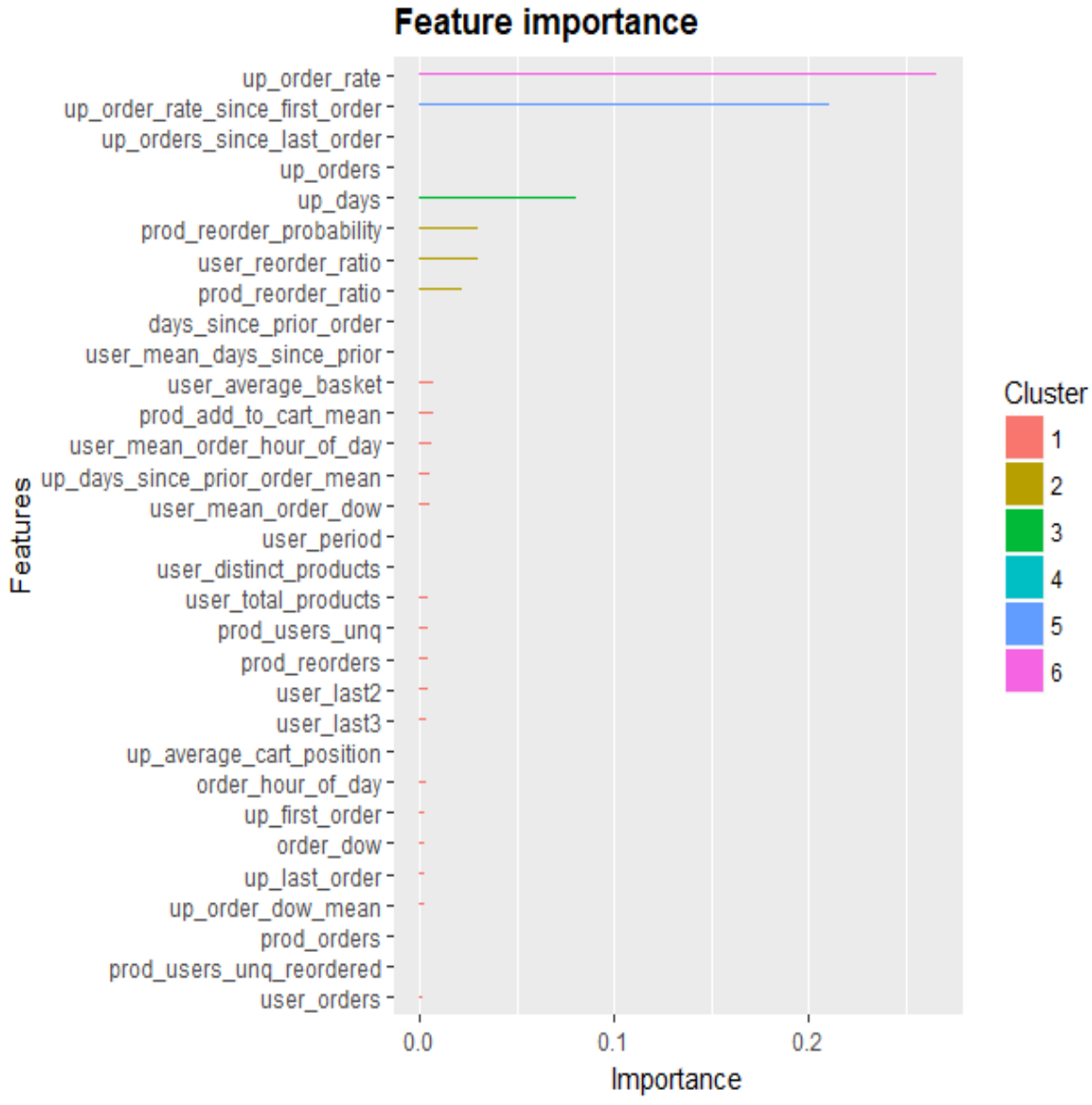
출력변수

3-5. Market Basket Regression의 실습

- 고객이 해당 상품을 다음 구매에 재구매 할 것인지 (고객, 상품), 고객, 상품 에 대한 입력변수를 사용하여 부스팅 모형으로 예측해보자.

- 또한 threshold를 0.2로 하여 고객마다의 precision, recall, f1-score를 구한 후, 고객 평균 f1-score를 구해보자.

3-5. Market Basket Regression의 실습



3-5. Market Basket Regression의 실습

힌트 1: dplyr 패키지의 함수들을 이용하여 고객별로 추천 상품 개수, 실제 구매 상품 개수, 추천된 상품 중 실제 구매된 상품 개수를 센다.

```
> table <- ts.mat %>% group_by(user_id) %>% summarize(den.rec = sum(reordered), den.pre = sum(fitted.y),
+                                                     nom = sum(reordered==1 & fitted.y==1))
> head(table)
# A tibble: 6 x 4
  user_id den.rec den.pre   nom
    <int>   <int>   <dbl> <int>
1      50       6      17     5
2      90       5       7     3
3     138       1       1     0
4     139       0       4     0
5     163       5       2     0
6     206       4       5     1
```

ts.mat은 고객 id 와 실제 구매여부(reordered), 그리고 예상 구매여부(fitted.y)가 저장된 데이터 프레임.

3-5. Market Basket Regression의 실습

힌트 2: Precision과 Recall의 분모가 0인 것에 대한 처리.

- 실제 구매 상품이 0개 이면 "None"이라는 상품을 산다고 가정, 추천된 상품이 0개이면 "None"이라는 상품을 추천한다고 가정한다.
  - 예) 실제 구매 상품이 "None" 이고 추천 상품이 "None" 이면 recall의 분모, precision의 분모, 분자가 0에서 1이 된다.

Reference

[1] Jannach, D., Zanker, M., Felfernig, A. and Fridrich G. *Recommender Systems: An Introduction.* Cambridge, 2011. pp.51-58.

[2] Friedman, J., Hastie, T., Simon, N., Qian, J. and Tibshirani, R. *Package 'glmnet'*, 2017.

[3] Instacart. *Instacart Market Basket Analysis | Kaggle* https://www.kaggle.com/c/instacart-market-basket-analysis, 2017.

[4] Package 'xgboost'

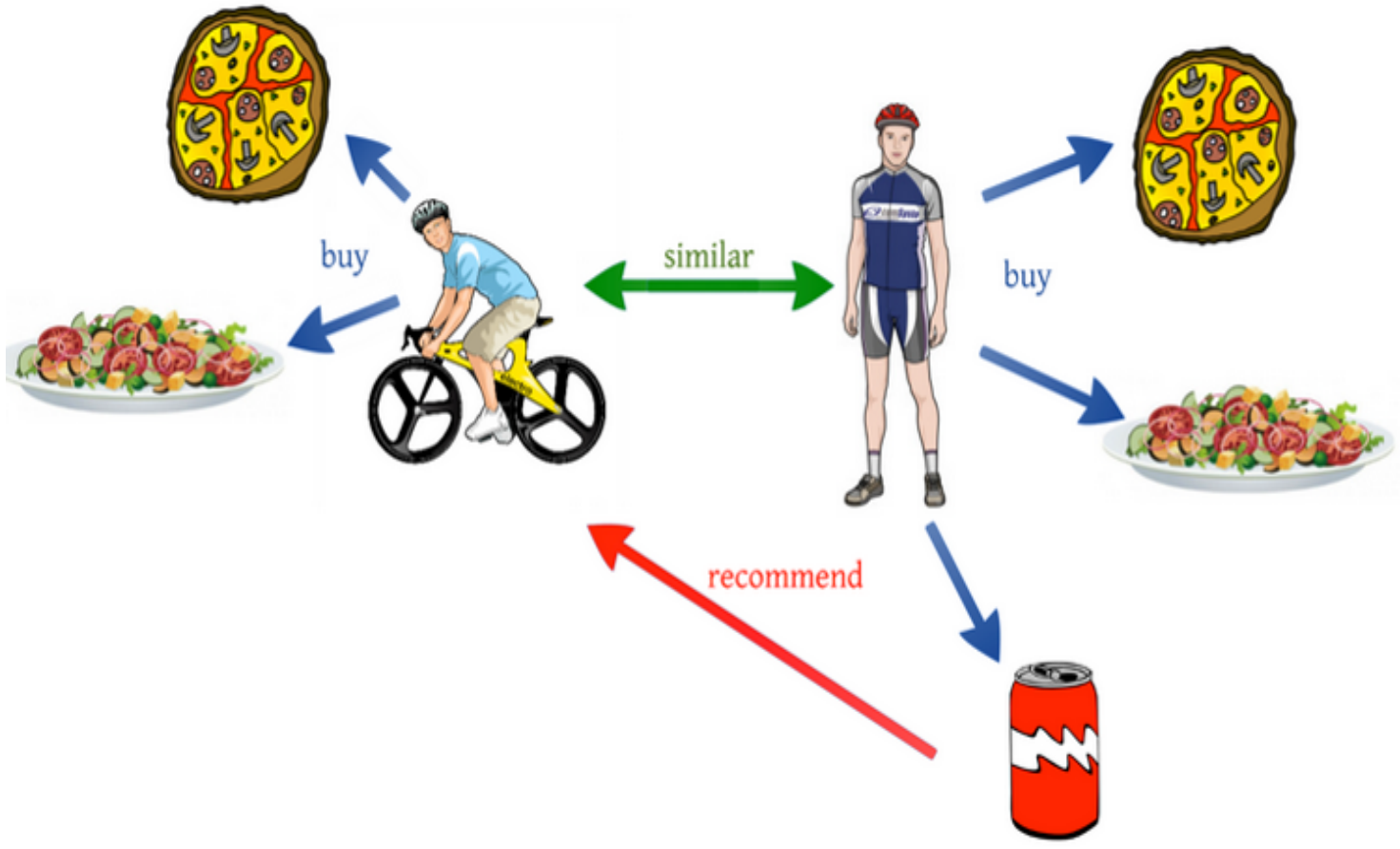
[5] Wickham, H., Francois, R., Henry, L., Muller, K. and RStudio. *Package 'dplyr'*, 2017.

4-2. 협력적 정화 방법 (Collaborative Filtering)

협력적 정화방법이란?

- 개인화된 추천을 위한 통계적 방법
  - 개인의 선호도와 과거 상품 구매이력 등을 분석하여 개인에게 최적인 상품을 추천.
- 기본 아이디어
  - 주어진 고객과 상품들에 대한 선호도가 비슷한 고객을 조사
  - 선호도가 비슷한 고객들이 좋아하는 상품 중에 주어진 고객이 모르고 있는 상품을 추천.
- 종류
  - 고객 중심의 협력적 정화방법(User-based)
  - 상품 중심의 협력적 정화방법(Item-based)

4-3. 고객 중심 협력적 정화 방법



4-3. 고객 중심 협력적 정화 방법

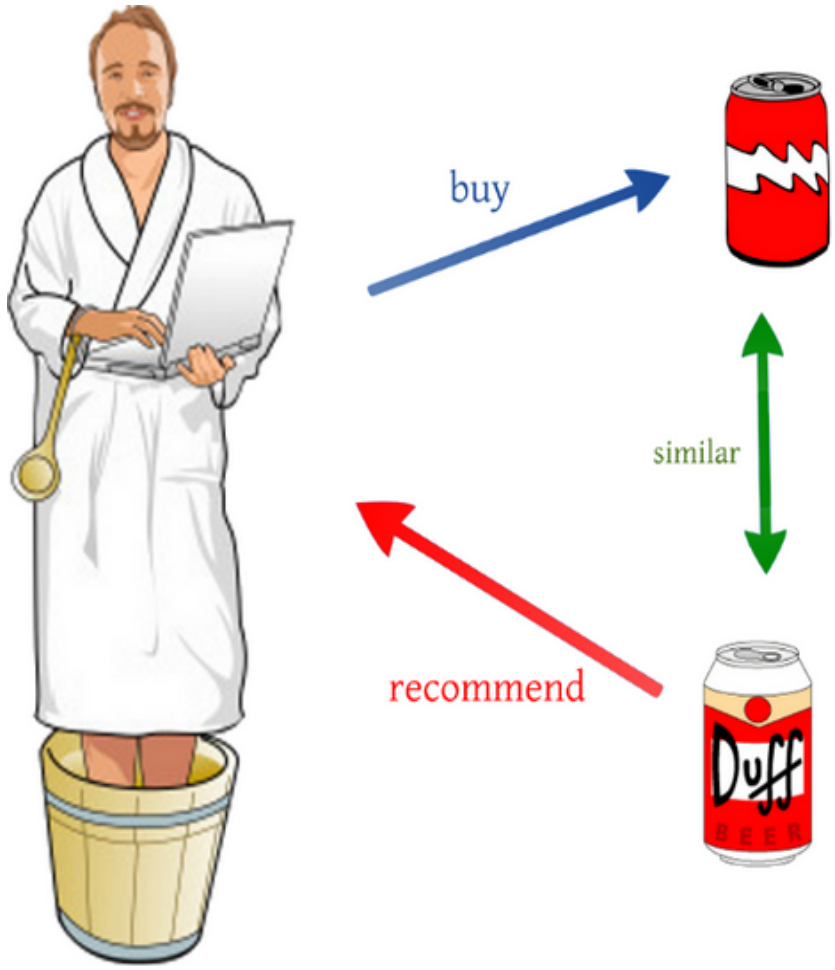
예제

- Eric과 유사성이 높은 사람이 Lucy와 Diane이라고 하자.
- Eric과 Lucy의 유사성척도 : 0.75
- Eric과 Diane의 유사성척도 : 0.15

	The Matrix	Titanic	Die Hard	Forrest Gump	Wall-E
John	5	1	2	2	
Lucy	1	5	2	5	5
Eric	2	?	3	5	4
Diane	4	3	5	3	

- $\hat{r} = \frac{0.75 \times 5 + 0.15 \times 3}{0.75 + 0.15} = 4.67$

4-4. 상품 중심 협력적 정화 방법



4-4. 상품 중심 협력적 정화 방법

예제

- Titanic과 유사성이 높은 영화가 Forest Gump와 Wall-E라 하자.
- Titanic과 Forest Gump의 유사성척도 : 0.85
- Titanic과 Wall-E의 유사성척도 : 0.75

	The Matrix	Titanic	Die Hard	Forrest Gump	Wall-E
John	5	1	2	2	
Lucy	1	5	2	5	5
Eric	2	?	3	5	4
Diane	4	3	5	3	

- $\hat{r} = \frac{0.85 \times 5 + 0.75 \times 4}{0.85 + 0.75} = 4.53$

4-5. 스케일 보정

- 고객마다 평가점수의 스케일이 다르다.
  - A고객은 대부분의 상품에 4점이나 5점을 주는 반면에 B 고객은 대부분의 상품에 1점이나 2점을 준다.
- 상품별로도 평가점수의 스케일이 다를 수 있다.

- 이러한 고객별/상품별 선호도의 스케일을 다음과 같이 보정한다.
  - $\mu_{ui} = \mu_0 + \mu_u^U + \mu_i^I$
  - $\hat{r_{ui}} = \mu_{ui} + \frac{\sum_{v \in \mathcal{R}_U^k(u,i)} (r_{vi} - \mu_{vi})}{|\mathcal{R}_U^k(u,i)|}$ 또는

    $\hat{r_{ui}} = \mu_{ui} + \frac{\sum_{v \in \mathcal{R}_U^k(u,i)} s^U(u,v)(r_{vi} - \mu_{vi})}{\sum_{v \in \mathcal{R}_U^k(u,i)} |s^U(u,v)|}$ 으로 선호도 예측
  - μ_{ui} 는 $\sum_{(u,i) \in \mathcal{R}} (r_{ui} - \mu_0 - \mu_u^U - \mu_i^I)^2$를 최소로 하여 추정함

4-6. 선호도 예측 평가방법

MAE vs RMSE

- Mean Absolute Error(MAE)

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|r_j - \hat{r}_j|$$

- Root Mean Square Error(RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(r_j - \hat{r}_j)^2}$$

여기서, r_j : 실제 평가 점수, $\hat{r}_j$: 추정된 평가 점수

4-8. 협력적 정화방법의 실습

Jester5k 자료

- R의 recommanderlab 패키지에 내장된 자료
- 1999년 4월 - 2003년 5월 중 평가된 평점들
- 사용자 수 : 5,000명, 상품 수 : 100개
- -10 - 10 사이의 평점이 존재

4-8. 협력적 정화방법의 실습

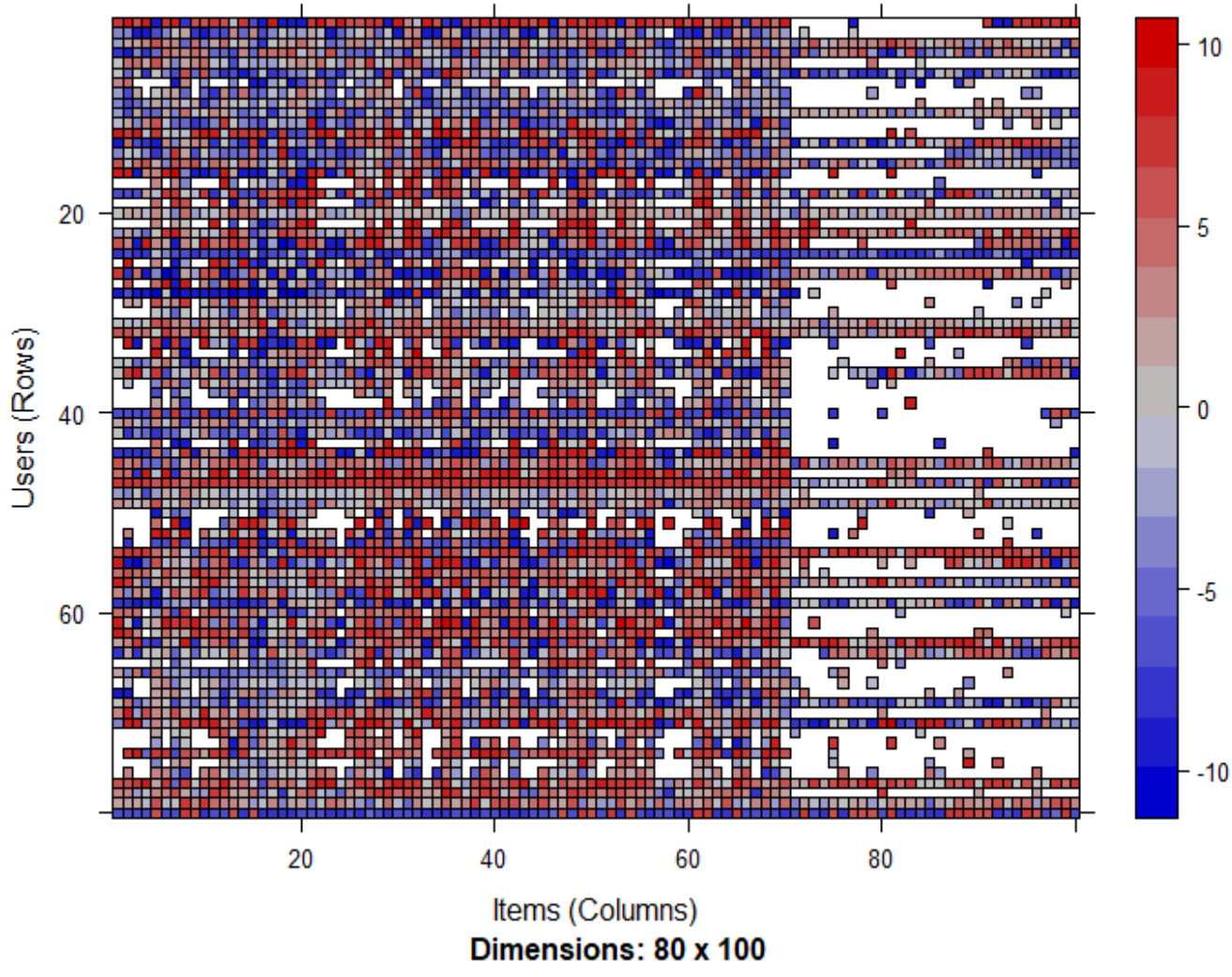
(1) 자료 불러오기 및 정리작업

```
> library(recommenderlab)
> data(Jester5k)
> head(as(Jester5k, "matrix"))
          j1    j2    j3    j4    j5    j6    j7    j8    j9   j10   j11   j12   j13   j14   j15   j16   j17   j18   j19
u2841   7.91  9.17  5.34  8.16 -8.74  7.14  8.88 -8.25  5.87  6.21  7.72  6.12 -0.73  7.77 -5.83 -8.88  8.98 -9.32 -9.08
u15547 -3.20 -3.50 -9.56 -8.74 -6.36 -3.30  0.78  2.18 -8.40 -8.79 -7.04 -6.02  3.35 -4.61  3.64 -6.41 -4.13 -0.15 -1.84
u15221 -1.70  1.21  1.55  2.77  5.58  3.06  2.72 -4.66  4.51 -3.06  2.33  3.93  0.05  2.38 -3.64 -7.72  0.97  2.04  1.94
u15573 -7.38 -8.93 -3.88 -7.23 -4.90  4.13  2.57  3.83  4.37  3.16 -4.90 -5.78 -5.83  2.52 -5.24  4.51  4.37  4.95  5.49
u21505  0.10  4.17  4.90  1.55  5.53  1.50 -3.79  1.94  3.59  4.81 -0.68 -0.97 -6.46 -0.34 -2.14 -2.04 -2.57 -0.15  2.43
u15994  0.83 -4.90  0.68 -7.18  0.34 -4.32 -6.17  6.12 -5.58  5.44 -4.85 -7.62 -6.65 -9.37 -6.07 -1.26 -7.43 -6.89 -3.64
```

- recommenderlab 라이브러리를 불러오면 Jester5k 자료를 바로 받을 수 있다
- Jester5k : 5,000 $\times$ 100 ratingmatrix 형태의 자료
- 자료의 기본 구조를 확인하려면 matrix형으로 변환하면 된다.
  - ✓ i번째 행 : i번째 사용자, j번째 열 : j번째 상품
  - ✓ (i,j)의 원소 : i번째 사용자가 j번째 상품에 매긴 평점

4-8. 협력적 정화방법의 실습

(1) 자료 불러오기 및 정리작업



4-8. 협력적 정화방법의 실습

(1) 자료 불러오기 및 정리작업

- 사용자1 - 사용자 1000의 평가자료의 30%를 랜덤하게 검증자료로 사용.

```
> n.user=1000; n.item=ncol(Jester5k)
> a <- as(Jester5k[1:n.user],"matrix")
> set.seed(123)
> subset=sample.int(sum(!is.na(a)),sum(!is.na(a))*.3)
> subset=sample.int(sum(!is.na(a)),sum(!is.na(a))*.3)
> subset=sort(subset)
> train = a; test =a
> train[!is.na(train)][subset] = NA; test[!is.na(test)][-subset] = NA
```

4-8. 협력적 정화방법의 실습

(2) 모형 적합

- 사용자1 - 사용자 1000의 평가자료의 학습자료를 바탕으로 사용자 기반 협력적 정화 추천알고리즘을 만들고자 한다.
- Recommender()
  - Recommenderlab 패키지에 포함된 함수
  - 입력하는 자료와 추천알고리즘을 지정
  - 자료 : ratingmatrix 형태만 입력 가능
  - 추천방식 : UBCF(user-based CF, 사용자기반), IBCF(item-based CF, 상품기반)을 주로 사용

```
> rtrain = as(train,"realRatingMatrix")
> r=Recommender(rtrain, method="UBCF")
```

4-8. 협력적 정화방법의 실습

(3) 모형 해석

- predict() 를 사용해서 사용자 1, 2에 대해 모형을 적용해보자.
- 전체 상품에 대한 예상 평점 계산
  - predict 함수 안에 type을 rating으로 지정
  - 이미 평점을 내린 항목에 대해서는 예상평점을 계산하지 않는다.

```
> pr=predict(r, rtrain[1:2,], type="ratings")
> as(pr, "matrix")
```

- 전체 상품 중 높은 평점이 기대되는 상품을 추천
  - predict 함수 안에 추천받을 상품의 개수를 입력
  - 예상 평점이 높은 순으로 결과를 보여준다.

```
> ptype=predict(r, rtrain[1:2,], n=5)
> as(ptype, "list")
```

4-8. 협력적 정화방법의 실습

(4) 결과

전체 상품에 대한 예상 평점 결과 이미 평가한 항목의 경우 NA로 표기된다

```
> pr=predict(r, rtrain[1:2,], type="ratings")
> as(pr, "matrix") #이미 평가한 자료의 경우 예측값을 주지 않는다
              j1        j2        j3        j4 j5 j6 j7        j8 j9 j10 j11       j12        j13
u2841         NA  4.720333 4.386004        NA NA NA NA 5.136567 NA   NA   NA 3.764987         NA
u15547 -3.265366 -4.139931        NA -6.256647 NA NA NA        NA NA   NA   NA        NA -3.292889
          j14       j15       j16       j17 j18 j19       j20       j21       j22       j23 j24
u2841  3.90231  3.782743 3.405311        NA   NA   NA  4.366890        NA 3.778064 4.005367   NA
u15547      NA -3.070682       NA -3.855951   NA   NA -3.843586 -2.729407        NA        NA   NA
             j25       j26 j27       j28       j29       j30 j31 j32 j33 j34       j35
u2841         NA        NA  NA        NA        NA  3.426336  NA  NA  NA  NA 5.743639
u15547 -2.127857 -2.417849  NA -0.740625 -1.656403 -5.479528  NA  NA  NA  NA        NA
             j36       j37 j38 j39 j40 j41 j42       j43 j44 j45 j46       j47 j48       j49
u2841         NA 3.301987  NA  NA  NA  NA  NA  3.720188  NA  NA  NA 3.839794  NA        NA
u15547 -0.7003748       NA  NA  NA  NA  NA  NA -7.839079  NA  NA  NA       NA  NA -1.331727
       j50 j51 j52       j53       j54       j55       j56 j57       j58       j59       j60
u2841   NA  NA  NA        NA 5.613841        NA  4.847810  NA        NA 3.234196        NA
u15547  NA  NA  NA -1.981252       NA -4.745778 -2.957346  NA -6.832037        NA -5.158318
             j61       j62       j63 j64       j65       j66 j67       j68 j69 j70       j71
u2841  4.608919  5.230829  4.685734  NA        NA        NA  NA 4.989884  NA  NA  3.764838
u15547       NA -1.564872 -2.540157  NA -1.213829 -2.104987  NA        NA  NA  NA -3.705984
             j72       j73       j74       j75       j76       j77       j78       j79
u2841  4.536993  3.732089  3.790887  3.774823  4.076120 3.923352  3.876145  3.876218
u15547       NA -2.982802 -3.644875 -2.909046 -2.932653        NA -2.945941 -3.478146
             j80       j81       j82       j83       j84       j85       j86       j87
u2841   3.548946  4.724597  4.081005  3.822898  4.555032  3.628251  4.297411  3.930296
u15547 -2.770954 -2.373534 -3.262728 -2.877906 -2.789100 -3.230916 -3.882870 -3.032505
             j88       j89       j90       j91       j92       j93       j94       j95
u2841   4.691537  4.507983  4.209081        NA        NA        NA        NA  3.773711
u15547 -3.064998 -2.349750 -2.970223 -3.077602 -3.045691 -3.110273 -3.160848 -3.285826
             j96       j97       j98       j99      j100
u2841         NA        NA        NA        NA        NA
u15547 -3.047358 -2.988154 -2.712941 -3.232245 -2.273629
```

4-8. 협력적 정화방법의 실습

(4) 결과

- 각 사용자에게 추천하는 상품들
  - 사용자 1 : 35, 54, 62, 8, 68번 상품 순으로 추천
  - 사용자 2 : 36, 28, 65, 49, 62번 상품 순으로 추천

```
> ptype=predict(r, rtrain[1:2,], n=5)
> as(ptype, "list")
$u2841
[1] "j35" "j54" "j62" "j8"  "j68"

$u15547
[1] "j36" "j28" "j65" "j49" "j62"
```

4-8. 협력적 정화방법의 실습

(4) 결과

- 검증자료를 사용해서 test RMSE를 구해보자.
- recommenderlab 패키지의 RMSE(true, predicted) 함수 사용.
  - true 에는 실제 점수 matrix를 입력
  - predicted 에는 예측 점수 matrix 입력.
  - default 로 NA는 제거하고 계산함.

```
> pr=predict(r, rtrain, type="ratings")
> pr=as(pr, "matrix")
> pr[pr>10]=10
> pr[pr<(-10)]=-10
> RMSE(test, pr)
[1] 4.423504
```

4-8. 협력적 정화방법의 실습

Jester5k 자료(스케일 보정)

- 앞에서 분석한 Jester5k 자료에 스케일 보정을 추가
- μ_0, μ_u^U, μ_i^I에 대한 추정이 우선되어야 한다.
- 능형회귀방법을 이용해 μ_0, μ_u^U, μ_i^I 를 추정

4-8. 협력적 정화방법의 실습

(1) 자료 가공

- 사용자 1 사용자 1000의 정보를 이용해 모형 구축
- 평점, 사용자, 상품이 각 열이 되는 행렬을 생성
  - 학습자료에 총 50,651건의 평점이 존재
- 평점을 사용자와 상품으로 능형회귀분석
  - 사용자와 상품은 명목형 변수이므로 더미 변수를 생성한 후 능형회귀분석을 진행

```
> dgmat=cbind(train[1:(n.user*n.item)], as.data.frame(cbind(rep(rownames(train), n.item),
+                                                           rep(colnames(train), each=n.user))))
> colnames(dgmat) <- c("rating","user","item")
> user = unique(dgmat$user); item = unique(dgmat$item)
> dgmat = dgmat[is.na(dgmat$rating)==F,]
> dummy = model.matrix(rating~user+item, dgmat)
> dummy = dummy[,-1]
```

4-8. 협력적 정화방법의 실습

(2) 능형회귀분석

- glmnet 패키지에 내장된 cv.glmnet, glmnet 함수 이용
- cv.glmnet 함수를 통해 λ 추정
- cv.glmnet 함수를 통해 얻은 λ 를 사용해서 μ_0, μ_u^U, μ_i^I를 추정.

```
> library(glmnet)
> set.seed(100)
> cv.lm = cv.glmnet(dummy, dgmat$rating, type.measure = "deviance", alpha=0)
> lm= glmnet(dummy, dgmat$rating, family="gaussian", lambda = cv.lm$lambda.min, alpha=0)
> head(coef(lm))
6 x 1 sparse Matrix of class "dgCMatrix"
                    s0
(Intercept)  0.7062611
useru10048  -0.2738687
useru10056  -1.8131211
useru10074   3.1735635
useru10093  -3.5458194
useru10118  -1.3836792
```

4-8. 협력적 정화방법의 실습

(3) 사용자기반 협력적 정화

- 기존 평점에서 능형 회귀분석을 통해 추정한 μ_0, μ_u^U, μ_i^I 를 빼줌
- Recommender 함수를 사용하기 위해 ratingmatrix 형태로 자료의 형식을 변경.

```
> dgmat$rating = dgmat$rating - (lm$a0 + dummy %*% lm$beta)
> user.index = match(dgmat$user, user); item.index = match(dgmat$item, item)
> mat=sparseMatrix(i=user.index, j=item.index, x=dgmat$rating)
> mat=as.matrix(mat) ; mat[mat==0]=NA
> colnames(mat)=item; rownames(mat)=user
> mat= as(mat, "realRatingMatrix")
```

4-8. 협력적 정화방법의 실습

(3) 사용자기반 협력적 정화

- 앞의 분석과 같은 방식으로 분석 진행
- 사용자 1, 2의 예상평점을 계산
- 예상평점으로 나온 결과에 앞에서 추정한 μ_0, μ_u^U, μ_i^I 값을 더해주어야 한다.

```
r1= Recommender(mat, method="UBCF")
pr1=predict(r1, mat, type="ratings")
pr1 = as(pr1, "matrix")
rownames(lm$beta) = gsub('user','', rownames(lm$beta)); rownames(lm$beta) = gsub('item', '', rownames(lm$beta))
item=as.character(item); user=as.character(user)

# 추정된 값들을 따로 저장
tmp.cf=data.frame(as.matrix(rownames(lm$beta)), as.matrix(lm$beta))
mu.0=lm$a0
mu.u=data.frame(user)
mu.i=data.frame(item)

library(dplyr)
colnames(tmp.cf) =c("user", "coef"); mu.u <- mu.u %>% left_join(tmp.cf, by="user")
colnames(tmp.cf) =c("item", "coef"); mu.i <- mu.i %>% left_join(tmp.cf, by="item")
mu.u[is.na(mu.u$coef),]$coef = 0 ; mu.i[is.na(mu.i$coef),]$coef = 0

scale.value=matrix(mu.0, nrow = length(user), ncol = length(item))
scale.value = apply(scale.value, 2, function(x) x+mu.u$coef )
scale.value = t(apply(scale.value, 1, function(x) x+mu.i$coef))

pr1.final =scale.value + pr1
```

4-8. 협력적 정화방법의 실습

(4) 결과 해석

- 각 사용자에게 추천하는 상품들
  - 사용자 1 : 89, 72, 62, 35, 61번 상품 순으로 추천
  - 사용자 2 : 89, 62, 66, 36, 21번 상품 순으로 추천

```
> pr1.final[1:2,]
             j1        j2        j3        j4 j5 j6 j7        j8 j9 j10 j11       j12        j13
u2841        NA  5.055073 4.521174        NA NA NA NA 3.281264 NA  NA  NA 4.836509         NA
u15547 -2.221673 -3.901848        NA -5.901753 NA NA NA        NA NA  NA  NA        NA -5.299743
            j14       j15       j16       j17 j18 j19       j20       j21       j22       j23
u2841  5.599529 0.5009126 0.5253619        NA  NA  NA  3.339047        NA 4.115163 3.834426
u15547       NA -4.8165225        NA -4.662158  NA  NA -4.114228 -1.192278        NA        NA
       j24       j25       j26 j27       j28       j29       j30 j31 j32 j33 j34       j35
u2841   NA        NA        NA  NA        NA        NA  3.800038  NA  NA  NA  NA 5.925427
u15547  NA -3.149103 -2.382929  NA -1.921464 -1.419505 -5.456244  NA  NA  NA  NA        NA
            j36       j37 j38 j39 j40 j41 j42       j43 j44 j45 j46       j47 j48       j49
u2841        NA 2.862853  NA  NA  NA  NA  NA  2.101035  NA  NA  NA 3.431042  NA        NA
u15547 -1.025025        NA  NA  NA  NA  NA  NA -6.170326  NA  NA  NA        NA  NA -1.622529
       j50 j51 j52       j53       j54       j55       j56 j57       j58       j59       j60
u2841   NA  NA  NA        NA 5.71395        NA  5.434330  NA        NA 3.379182        NA
u15547  NA  NA  NA -1.584959        NA -4.027165 -3.149996  NA -7.877757        NA -4.030279
            j61       j62       j63 j64       j65       j66 j67       j68 j69 j70       j71
u2841  5.759909  6.0355764  4.195957  NA        NA        NA  NA 5.49558  NA  NA  2.905021
u15547       NA -0.9655176 -3.629876  NA -1.805994 -0.9748725  NA       NA  NA  NA -5.157923
            j72       j73       j74       j75       j76       j77       j78       j79
u2841  6.177687  4.238994  1.193265  2.889026  5.244365 3.896869  4.660105  3.119167
u15547       NA -2.903657 -5.973960 -4.402387 -1.765295        NA -2.625105 -4.514178
            j80       j81       j82       j83       j84       j85       j86       j87
u2841   3.786701  5.175305  3.837385  4.815744  3.880106  3.716081  3.487323  4.667864
u15547 -3.269868 -1.687689 -3.697503 -2.890367 -3.385145 -3.204015 -3.962208 -2.346576
            j88       j89       j90       j91       j92       j93       j94       j95
u2841   5.082323  6.4990925  3.948714        NA        NA        NA        NA  3.626943
u15547 -2.371100 -0.1600834 -3.646533 -2.620886 -3.005147 -2.532742 -3.234513 -3.670279
            j96       j97       j98       j99      j100
u2841        NA        NA        NA        NA        NA
u15547 -2.42025 -3.022222 -3.799956 -3.871319 -2.539423
```

4-8. 협력적 정화방법의 실습

(4) 결과 해석

- 검증자료를 사용해서 test RMSE를 구해보자.
  - 이전 방법과 비교했을 때 검증자료의 RMSE 가 더 큰 값을 갖는다.

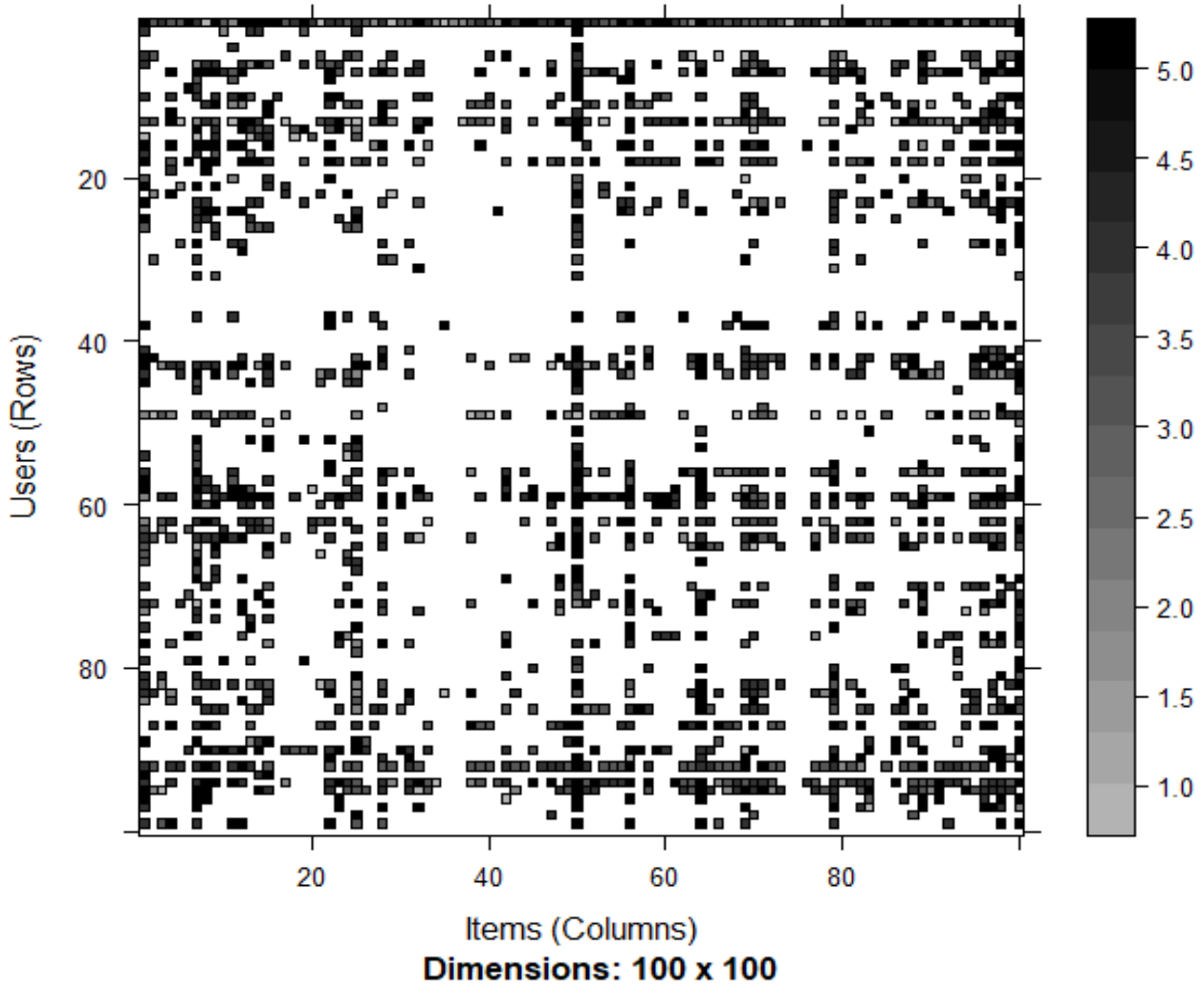
```
> pr1.final[pr1.final>10]=10
> pr1.final[pr1.final<(-10)]=-10
> RMSE(test,pr1.final)
[1] 4.440975
```

4-8. 협력적 정화방법의 실습

직접 해보기(MovieLense 자료)

- R의 recommanderlab 패키지에 내장된 자료
- MovieLens 웹 사이트 (movielen.umn.edu)에서 1997년 9월 - 1998년 4월까지 취합된 평점
- 1-5 scale 의 평점 100,000개 존재
- 사용자 : 943명, 영화 : 1,664편
- data("MovieLense") 함수로 호출 가능

4-8. 협력적 정화방법의 실습



Reference

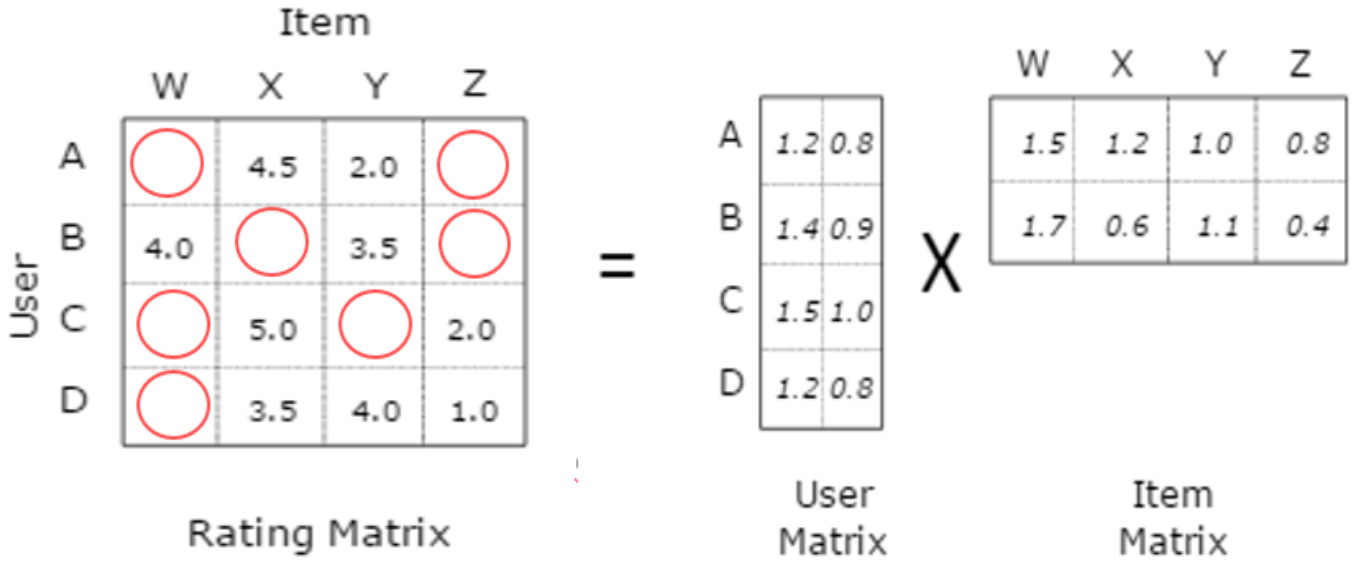
[1] Jannach, D., Zanker, M., Felfernig, A. and Fridrich G. *Recommender Systems: An Introduction.* Cambridge, 2011. pp.51-58, 13-50

[2] Ricci, F., Rokach, L., Shapira, B., and Kantor, P.B., eds. : *Recommender Systems Handbook.* Springer, 2010. pp.114-129, 148-150, 169-173

[3] Hahsler, M. *recommenderlab: A Framework for Developing and Testing Recommendation Algorithms*, 2017.

5-1. 행렬분해 방법(Matrix Factorization)

- 행렬 채워넣기



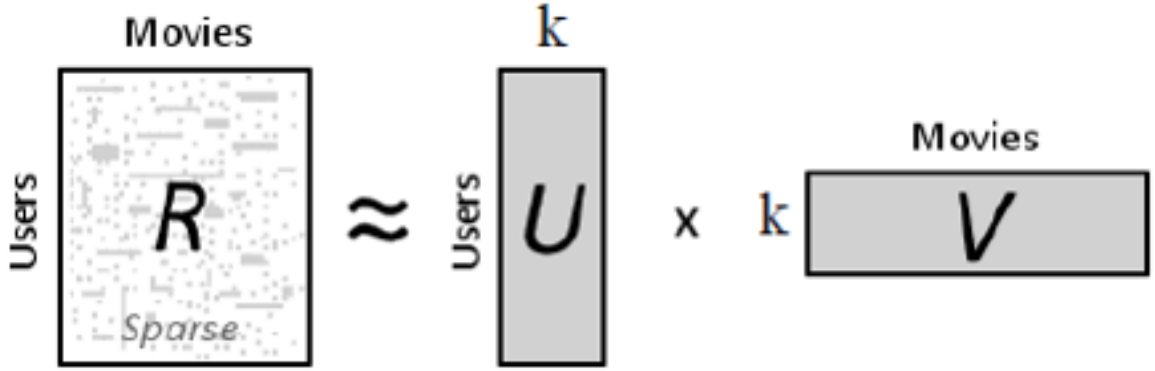
- 잠재 요인 모델(Latent factor model)을 적용한 행렬 분해 방법 이용.

5-1. 행렬분해 방법(Matrix Factorization)

- $\hat{r_{ui}} = \Phi_u^{U\prime} \Phi_i^I$으로 평점을 예측.
- 좀 더 일반적으로, 다음과 같이 예측하기도 함.

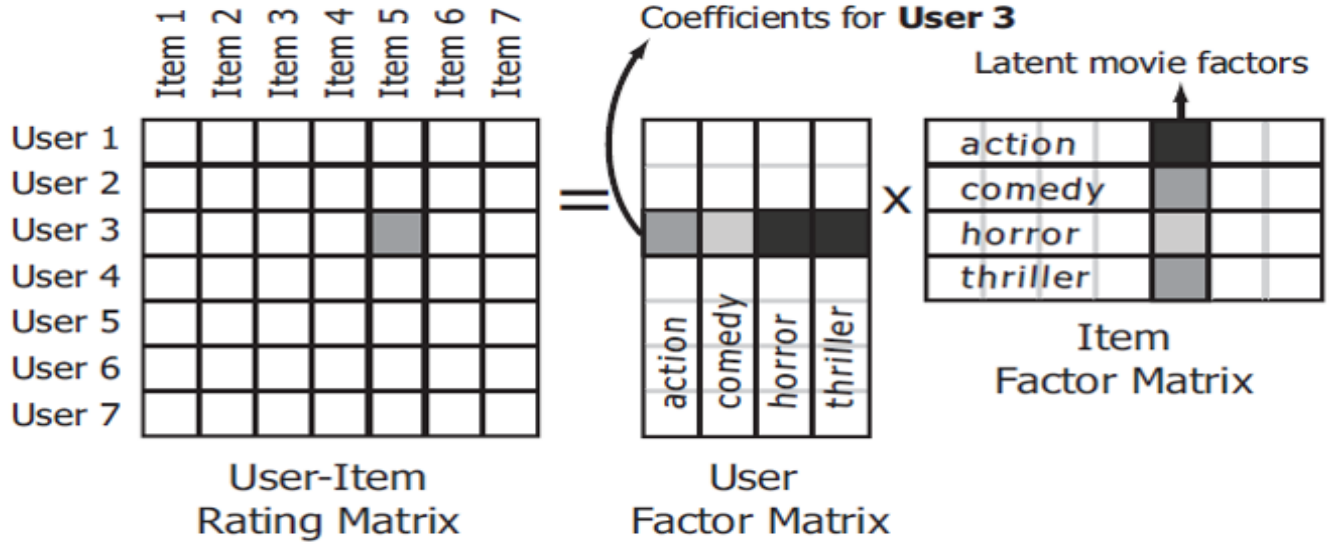
$$\hat{r_{ui}} = \mu_{ui} + \Phi_u^{U\prime} \Phi_i^I$$

여기서 $\mu_{ui} = \mu_0 + \mu_u^U + \mu_i^I$.



5-1. 행렬분해 방법(Matrix Factorization)

예제



5-3. Context aware 추천방법

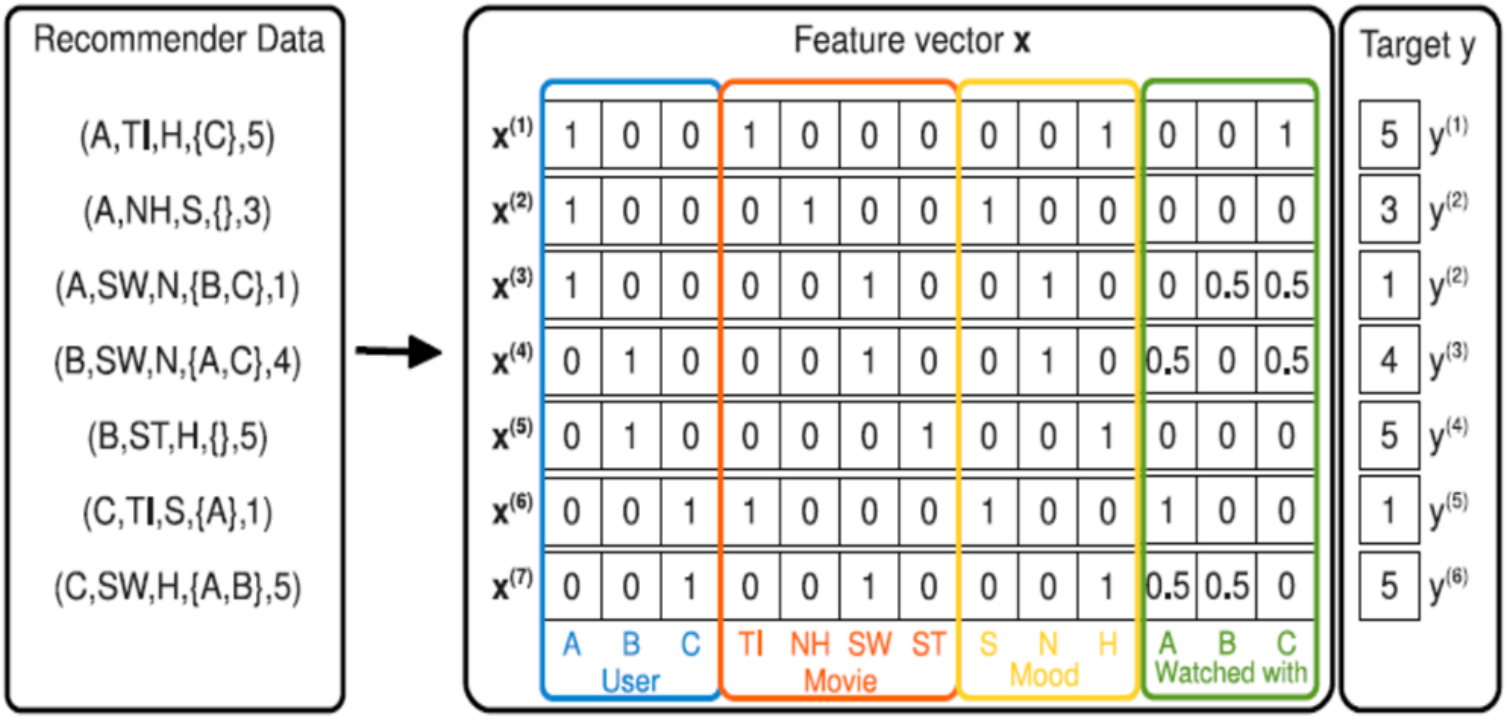
Factorization machine 추천 시스템

Ex.

- 2가지의 context가 존재
  (고객의 기분 상태와 함께 본 사람의 weight 벡터)
- $U = \{\text{Alice, Bob, Charlie}\}$
- $I = \{\text{Titanic, Notting Hill, StarWars, StarTrek}\}$
- $C_1 = \{\text{Sad, Normal, Happy}\}$
- C_2 : 해당 고객과 함께 본 사람의 weight 벡터

5-3. Context aware 추천방법

Factorization machine 추천 시스템



5-3. Context aware 추천방법

- Factorization machine

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

where $x \in \mathbb{R}^n$, $w_{ij} =< v_i, v_j >= \sum_{k=1}^{K} v_{ik} v_{jk}$ and K is the hyperparameter of the dimensionality of the factorization.

- 벌점화 함수가 추가된 목표 함수를 최소화 하는 $\theta = (w_0, W, V)'$ 를 추정.

$$\min \sum_{(x,y) \in S} (\hat{y}(x) - y)^2 + J_\lambda(\theta)$$

여기서 S는 observed data set.

5-4. 행렬분해 방법의 실습

Jester5k 자료

- 협력적 정화 기법과 마찬가지로, Jester5k data 사용
- 업로드된 matfact 패키지 사용
  - \R\R-3.2.2\library 에 압축 풀기
  - 또는 tools -> install.packages->
    install from: "Package Archive File(.zip; tar.gz)" 선택 ->
    Package archive: zip 파일 경로에서 matfact.zip 선택 ->
    install

5-4. 행렬분해 방법의 실습

- matfact()
  - matfact 패키지에 포함된 함수
  - 입력하는 자료와 벌점화 모수, 행렬분해 차원, 반복횟수를 지정
  - 자료 : matrix 형태 입력 가능
  - 기본 설정값 :
    (벌점화 모수, 분해 차원, 반복 횟수) = (0.2, 5, 100)

5-4. 행렬분해 방법의 실습

(1) 모형 적합

- 앞의 협력적 정화기법과 비교하기 위해, 마찬가지로 사용자 1-1000의 평점정보의 학습자료를 바탕으로 행렬분해기법을 적용

```
> library(matfact)
> mf = matfact(train)
> pred = mf$P %*% t(mf$Q)
> colnames(pred)=item ; rownames(pred)=user
>
> head(pred[,1:10])
              j1        j2        j3         j4         j5        j6         j7         j8          j9         j10
u2841   3.47299097  5.171799  6.083851  8.2108792  2.5793112  4.953541 -0.4831361  0.6947555  7.59102308  3.68738783
u15547 -3.77973106 -4.505043 -6.076120 -6.6760374 -0.3735404 -1.939386 -0.2675582 -2.4498265 -7.76971135 -4.43070862
u15221  0.32945166  1.607225  0.337589  0.4945241  2.0203230  2.962595 -1.1347846 -1.7965840 -0.88731881  0.07043865
u15573  0.32281728 -2.757928 -1.437332 -4.5976469 -2.4429121 -1.900897  1.1791064  1.8992427 -1.10616265  0.62072602
u21505  0.66603372  1.687600  0.727674  1.2643956  1.8670623  2.499960 -0.3463237 -1.0086666  0.06168396  0.30434832
u15994  0.03796293 -2.709482 -1.906574 -7.9204660 -4.6172598 -4.566587 -0.3409255  1.7850986 -3.32561153  1.23727437
```

분해된 행렬을 곱함으로써 1000 $\times$ 100 예상 평점 행렬 pred를 얻음

5-4. 행렬분해 방법의 실습

(2) 결과 해석

- 사용자 1, 2가 평점을 내리지 않은 상품들에 대해 예상 평점을 계산
- 각 사용자에게 추천하는 상품들
  - 사용자 1 : 30, 43, 59, 23, 79번 상품 순으로 추천
  - 사용자 2 : 89, 93, 78, 76, 83번 상품 순으로 추천

```
> i1 = is.na(train[1,])
> i2 = is.na(train[2,])
>
> sort(pred[1,i1==T],decreasing=T)
       j30        j43        j59        j23        j79         j3        j74        j12        j22
9.19354169 8.50875752 6.82089059 6.69426387 6.58255209 6.08385113 5.90710806 5.77996875 5.67957436
       j63         j2        j14        j80        j37        j47        j84        j77        j20
5.46598591 5.17179901 5.07081204 5.02449186 4.89440689 4.54031846 4.52481714 4.49125904 4.47922672
       j73        j85        j75        j86        j71        j61        j16        j90        j68
4.40341642 4.35773002 4.21186036 4.13735103 3.50843888 3.42145674 3.36946293 3.09610458 2.58217506
       j87        j82        j56        j81        j78        j95        j54        j88        j62
2.53152624 2.41404811 2.40859252 2.27161161 2.23554318 2.16057245 2.10146776 1.98051047 1.39111693
       j83        j35         j8        j72        j15        j76        j89
1.04950948 0.85458777 0.69475554 0.62480300 0.24405762 -0.03002685 -1.86354901
> sort(pred[2,i2==T],decreasing=T)
       j89        j93        j78        j76        j83        j53        j62        j97
4.096008169 2.276181264 2.234625178 2.204166132 1.813875631 1.038616694 1.006858544 0.980530553
       j36        j81        j88        j73        j29        j96        j87        j66
0.857986719 0.547728197 0.547338571 0.545216745 0.477835154 0.401310961 0.008776939 -0.130378411
       j49        j91        j65        j80        j82        j95        j56        j98
-0.266657367 -0.373001831 -0.714181244 -0.733167593 -0.805274942 -0.833454957 -0.838142950 -0.907481746
       j92       j100        j75        j90        j94        j99        j21        j71
-0.928699094 -1.173888356 -1.212889982 -1.270332961 -1.283471987 -1.391766629 -1.505391148 -1.599962493
       j28        j17        j15        j84        j85        j79        j26        j20
-1.725093006 -1.748064945 -1.887679628 -1.888085169 -2.066646092 -2.194565354 -2.646896678 -2.716124957
       j86        j74        j13         j1        j63        j25         j2        j55
-3.049573498 -3.293809825 -3.701613209 -3.779731057 -3.871283763 -4.402585570 -4.505042631 -5.922341884
        j4        j60        j30        j43        j58
-6.676037352 -6.957270493 -6.997319175 -7.219138490 -8.417116093
```

4-8. 협력적 정화방법의 실습

(2) 결과 해석

- 검증자료를 사용해서 test RMSE를 구해보자.
  - 고객 기반 추천 방법과 비교했을 때 검증자료의 RMSE 가 더 작은 값을 갖는다.

```
> pred[pred>10]=10
> pred[pred<(-10)]=-10
> RMSE(test,pred)
[1] 4.324519
```

- 학습자료를 사용해서 train RMSE를 구해보자.
  - 행렬분해에서는 평점을 내린 상품들에 대해서도 예상 평점을 계산하기 때문에 train RMSE를 구할 수 있음

```
> RMSE(train,pred)
[1] 3.791257
```

5-4. 행렬분해 방법의 실습

Jester5k 자료(스케일 보정)

(1) 모형 적합

- 협력적 정화 알고리즘과 마찬가지로, 행렬분해 기법에서도 스케일 보정을 사용가능
- 스케일 보정 -> 행렬분해 기법 적용 -> 스케일 재보정

```
> mat1=as(mat,"matrix")
> mf1 = matfact(mat1)
> pred1 = mf1$P %*% t(mf1$Q)
> colnames(pred1)=item ; rownames(pred1)=user
> pred1.final=pred1[1:2,]+scale.value
```

5-4. 행렬분해 방법의 실습

(2) 결과 해석

- 사용자 1, 2가 평점을 내리지 않은 상품들에 대해 예상 평점을 계산
- 각 사용자에게 추천하는 상품들
  - 사용자 1 : 3, 30, 85, 23, 43 번 상품 순으로 추천
  - 사용자 2 : 89, 93, 36, 76, 53 번 상품 순으로 추천

```
> sort(pred1.final[1,i1==T],decreasing=T)
       j3       j30       j85       j23       j43       j59       j84       j77       j80       j22       j79
13.0655101 12.7951682 11.4832778  9.9704068  9.8268684  8.4767443  7.9602207  7.5905113  7.5657228  7.5230870  7.3366230
       j63       j86       j90       j73       j12       j88       j47       j54       j78        j2       j20
 7.1878214  7.1599621  6.8595826  6.5409505  5.8676120  5.4741676  4.7729502  4.5911541  4.5747875  4.5478327  4.5462695
       j82       j81       j68       j61       j37       j62       j14       j71       j87       j95       j89
 4.4338700  4.1909004  4.1082032  4.0877313  3.9164063  3.4035151  3.1524356  2.9852929  2.6891217  2.0734433  2.0269562
       j72       j75        j8       j74       j83       j56       j35       j76       j15       j16
 1.9963829  1.9767148  1.4006602  1.1375989  0.9836948 -0.6420206 -0.6847232 -1.5967852 -3.5604327 -4.5020416
>
> sort(pred1.final[2,i2==T],decreasing=T)
       j89       j93       j36       j76       j53       j78       j83       j62       j29       j66       j81
 2.9415124  1.3795222  1.0872828  1.0458136  0.7415674  0.6106290  0.5467737  0.4433534  0.3766477  0.1580539 -0.2150629
       j88       j96       j49       j97       j87       j91       j73       j56       j65       j92       j21
-0.4064502 -0.4719742 -0.6503573 -0.6951865 -0.7181233 -0.7826391 -0.9988168 -1.2357650 -1.2818386 -1.2983619 -1.6627324
       j80       j82       j95       j28       j94      j100       j15       j90       j98       j75       j99
-1.7761062 -1.7823865 -1.8476343 -2.0700527 -2.2054530 -2.3281606 -2.4976803 -2.5276648 -2.6302441 -2.6701970 -2.7409104
       j26       j17       j79       j71       j85       j84        j1       j13       j20       j86       j63
-2.8124672 -2.8984713 -3.1638863 -3.5007134 -3.5945489 -3.8044119 -3.8251926 -4.2911910 -4.3108679 -4.3599586 -4.6694884
       j25        j2       j74       j55       j60       j43       j30        j4       j58
-4.8381544 -5.2709497 -5.5899810 -5.7722003 -6.9939470 -7.1385249 -7.2296725 -7.2912326 -9.4754507
```

4-8. 협력적 정화방법의 실습

(2) 결과 해석

- test RMSE, train RMSE를 구해보자.
  - 스케일 보정 전과 비교했을 때 train RMSE는 더 작은 값을 갖지만, test RMSE는 더 큰 값을 갖는다.

```
> pred1.final[pred1.final>10]=10
> pred1.final[pred1.final<(-10)]=-10
> RMSE(test,pred1.final)
[1] 4.345545
>
> RMSE(train, pred1.final)
[1] 3.714567
```

5-4. 행렬분해 방법의 실습

직접 해보기(MovieLense 자료)

- 4-8절에서와 마찬가지로 Recommenderlab 패키지의 MovieLense 자료로 행렬분해 모형을 적합해보자.

5-5. Factorization machine의 실습

영화 평점 자료(Movielense, ml100k.csv)

- recommender lab 패키지에 내장된 자료와 거의 같지만, 시간에 대한 변수가 추가됨.
- 사용자 943명, 영화 1,682편에 대한 100,000개의 1-5의 평점 자료.

```
> head(ml100k)
  user item rating      time
1  196  242      3 881250949
2  186  302      3 891717742
3   22  377      1 878887116
4  244   51      2 880606923
5  166  346      1 886397596
6  298  474      4 884182806
```

네번째 열은 사용자가 영화를 평가한 UNIX 시간임.

5-5. Factorization machine의 실습

(1) 자료 정리 작업

- as.POSIXlt 함수를 사용하여 UNIX 시간을 요일로 변환.
- 요일을 context 정보로 하여 변수로 사용.
- 사용자, 상품, 요일이 각 열이 되는 행렬 생성
- output 벡터는 따로 저장한다.

```
> library(Matrix)
> user=ml100k[,1]
> items=ml100k[,2]+max(user)
> wdays=(as.POSIXlt(ml100k[,4],origin="1970-01-01")$wday+1)+max(items)
>
> data=sparseMatrix(i=rep(1:nrow(ml100k),3),j=c(user,items,wdays),giveCsparse=F)
> target=ml100k[,3]
```

5-5. Factorization machine의 실습

(1) 자료 정리 작업

- 자료의 20%를 랜덤하게 뽑아 검증자료로 사용.

```
> set.seed(123)
> subset=sample.int(nrow(data),nrow(data)*.2)
> subset=sort(subset)
> data.train=data[-subset,]
> data.test=data[subset,]
> target.train=target[-subset]
> target.test=target[subset]
```

5-5. Factorization machine의 실습

(2) 모형 적합

- FactoRizationMachines 패키지의 FM.train 함수 사용.
- FM.train(data,target,factors=c(1,10),iter=100,regular=0)
  - 2차 교호작용의 factorization machine 을 학습시키는 함수.
  - factor=c(a,b) :
    1차항의 weight 사용 설정하면 $a = 1$, 아니면 0,
    b는 2차항의 Factorization dimension (5-3절에서 K)
  - iter : iteration 횟수 설정
  - regular : 벌점화 모수 설정

```
> library(FactoRizationMachines)
> set.seed(1)
> model=FM.train(data.train,target.train,regular=0.1, c(1,10), iter=200)
```

5-5. Factorization machine의 실습

(3) 결과

- predict 함수를 이용하여 검증 자료의 평점 예측

```
> pre=predict(model,data.test)
> head(pre)
[1] 2.138973 4.192368 3.329251 3.143009 4.254716 4.015136
```

- RMSE 를 구해본다.

```
> sqrt(mean((pre-target.test)^2))
[1] 0.9123031
```

5-5. Factorization machine의 실습

직접 해보기(Restaurant 평점 자료 (RCdata.csv))

- 고객 138명, 식당 130개
- 0,1,2의 평점 1161개 존재.
- 고객에 대한 아래와 같은 세가지 명목형 변수를 context로 사용하여 Factorization machine 모형을 적합해보자.
  - drink-level : abstemious, casual drinker, social drinker
  - interest : eco-friendly, none, retro, technology, variety
  - personality : conformist, hard-worker, hunter-ostentatious, thrifty-protector

Reference

[1] Koren, Y., Bell, R., and Volinsky, C. *Matrix Factorization Techniques for Recommender Systems*, IEEE Computer, 2009.

[2] Ricci, F., Rokach, L., Shapira, B., and Kantor, P.B., eds. : *Recommender Systems Handbook.* Springer, 2010. pp.151-154.

[3] Karatzoglou, A., Amatriain, X., Baltrunas, L., and Oliver, N. *Multiverse Recommendation: N-dimensional Tensor Factorization for Context-aware Collaborative Filtering*, RecSys, 2010.

[4] Rendle, S. *Factorization Machines*, ICDM, 2010.

[5] Knoll, J. *Package 'FactoRizationMachines'*, 2017.

[6] Grouplens. *MovieLense Dataset.*, https://grouplens.org/datasets/movielens/.

[7] Vargas-Govea, B., GonzAiles-Serna, J. G. and Ponce-MedellAn, R. *Effects of relevant contextual features in the performance of a restaurant recommender system.*, RecSys, 2011.