이학석사 학위논문

그룹 벌점 함수를 활용한 순위 모형 분석

Group Ranking Lasso

2017년 2월

서울대학교 대학원

통계학과

조 수 환

국문초록

순위 데이터는 어떠한 기준에 따라 순위를 매기고자 할 때 발생한다. 일상생활 속에서도 순위 데이터를 쉽게 접할 수 있는 만큼 순위 추정 알고리즘은 스포츠, 통계 등 다양한 분야에서 활용할 수 있어 그 관심도가 높다. 순위 추정 모형으로는 Luce 모형이 잘 알려져 있다. Luce 모형에 대해 pairwise penalty 함수를 이용한 정규화 방법을 적용하면 차이가 적은 항목들끼리 순위가 묶여서 나타난다는 것이 알려져 있다. 이렇게 추정된 sparse 모형은 순위의 해석을 용이하게 하고 예측 성능을 향상시킨다. 본 연구에서는 비교할 항목이 여러 그룹으로 나누어져 있는 경우에 대해 Luce 모형에 group penalty 함수를 이용한 정규화 방법을 적용함으로써 그룹 단위의 순위를 비교하였다. group penalty를 적용한 모형의 모수를 ADMM 알고리즘과 cyclic MM 알고리즘을 이용하여 효율적으로 추정하였다. 제안한 모형을 2015년도 미국 프로 미식축구 정규시즌 경기 데이터에 적용하여 분석하였다.

주요어 : Luce 모형, 순위 모형, Pairwise penalty, Group penalty

학　번 : 2015-20313

Contents

List of Tables

List of Figures

Chapter 1

서론

순위 데이터는 어떠한 기준에 따라 순위를 매기고자 할 때 발생한다. 예를 들면, 스포츠 경기 결과에 의한 팀 순위나 관람객들의 평점에 따른 영화 순위 등이 있다. 이처럼 일상생활 속에서도 순위 데이터를 쉽게 접할 수 있는 만큼 순위 추정 알고리즘은 스포츠, 통계 등 다양한 분야에서 활용할 수 있어 그 관심도가 높다. 순위 추정 모형으로는 Luce 모형[8]이 잘 알려져 있다. 원래 Luce 모형은 여러 항목에 대한 순차적 선택 과정을 확률을 설명하기 위한 모형이었지만, Bradley-Terry 모형[3]과 Thurstonian 모형[10] 등에 의해 알려지게 된 후 Luce 모형으로부터 다양한 순위 모형들이 유도되었다.

최근의 연구들은 순위의 그룹화를 목표로 순위 모형에 대한 정규화 방법에 초점을 맞추고 있다. 순위 모형에서 비교하는 항목 간의 차이가 미미한 경우 그들을 하나로 묶기 위해 쌍별(pairwise) penalty 함수를 이용해 정규화한다[11]. 실제로 이러한 penalty 함수를 적용할 경우, 유효한 모수의 개수가 줄어들어 모형이 sparse 구조가 된다. 이러한 sparsity는 기존의 방법에 비해 추정된 모형의 해석을 용이하게 하고 예측 성능을 향상시킨다[9]. 그러나 대부분의 모형이 Bradley-Terry 모형과 같이 쌍별 비교로 유도된 모형으로 한정되어 있으며, 모

수 추정 시 매우 많은 양의 계산이 필요하다는 문제점이 있다. 특히, 비교하는 항목의 수가 많아질 경우 Newton-Raphson과 같은 계산 알고리즘을 이용하면 계산 결과 또한 불안정해진다. 이러한 단점을 극복하기 위해 [5]에서는 새로운 정규화 방법을 소개하였다. [5]의 모형이 갖는 차별성은 Luce 모형의 가능도 함수에 바로 정규화 방법을 적용하였다는 것과 양의 모수에 대해 모수들 간의 차이에 대한 penalty 함수를 부과한다는 것이다. 이전의 방법들은 쌍별 비교를 기반으로 하고 있으며, 모수의 로그값 간의 차이에 대해 penalty를 부과하고 있다. 또한, [5]의 penalty 함수를 이용하면 모수 추정 시 효율적인 계산 알고리즘을 활용할 수 있게 된다. 이전의 모형에서는 Newton-Raphson 방법을 이용하므로 Hessian matirx의 역행렬을 계산하는 과정이 필요했지만, 새로운 penalty 함수를 사용하면 간단한 제곱근 계산만으로 모수를 추정할 수 있어 훨씬 빠르고 안정적인 계산을 할 수 있다.

본 연구에서도 Luce 모형의 가능도함수에 대해 직접 정규화 방법을 이용할 것이다. 그러나 지금까지의 모형이 전체 모수에 대해 동일한 penalty를 적용하여 각 항목의 순위를 추정하였다면, 우리는 비교할 항목이 여러 그룹으로 나뉜 경우에 대한 그룹 단위의 순위 비교를 목표로 한다. 이를 위해 그룹 내, 그리고 그룹 간 비교에 따라 penalty 함수를 분리하여 적용한다. 그리고 제안한 방법을 2015년도 미국 프로 미식축구 리그(National Football League) 정규 시즌 데이터에 적용하여 결과를 분석한다.

논문의 구성은 다음과 같다. 2장에서는 Luce 모형에 대한 전반적인 개요와 그룹 단위 비교를 위한 정규화 방법을 소개하겠다. 3장에서는 제안한 모형의 모수를 추정하기 위한 최적화 알고리즘을 다룰 것이다. 4장은 제안한 모형을 NFL 2015 경기 데이터에 적용한 결과이며, 마지막 5장은 결론과 제언으로 마무리한다. 부록에는 4장에서 NFL 2015 경기 데이터를 분석한 R 코드를 첨부하였다.

Chapter 2

Luce model

Luce 모형은 여러 항목 중 매번 한 개의 항목을 선택하는 순차적 선택 과정에 대한 확률 모형이다. 여기서 선호도가 높은 항목 순서대로 선택한다고 가정한다. 전체 항목의 집합을 $S = \{1, \cdots, p\}$, i번째로 선택된 항목을 $j_i \in S$라 하면 i번째 선택 시 선택 가능한 항목들의 집합은 $S_i = S - \{j_1, \cdots, j_{i-1}\}$로 나타낼 수 있다. Luce 모형의 모수를 u_j, $j = 1, \cdots, p$라 하면 S_i 중 j_i를 선택하는 확률은 다음과 같이 정의된다. 이때 u_j는 모두 양의 값을 갖는다.

$$P(j_i \mid S_i) = \frac{u_{j_i}}{\sum_{j \in R_i} u_j} \tag{2.1}$$

Luce 모형에서 $(j_1 \to \cdots \to j_p)$의 순서로 선택이 발생한 사건의 확률은 다음과 같이 정의된다.

$$P(j_1 \to \cdots \to j_p) = \prod_{i=1}^{p-1} \frac{u_{j_i}}{\sum_{m=i}^{p} u_{j_m}} \tag{2.2}$$

여기서 두 개의 항목에 대한 비교 시 $P(j_1 \to j_2) = u_{j_i}/(u_{j_1} + u_{j_2})$임을 알 수 있다. 이는 Bradley-Terry 모형과 같은 확률식으로, 이를 통해 Luce 모형을 Bradley-Terry 모형의 확장된 형태로 이해할 수 있다.

Luce 모형은 순열(permutation)과도 관련이 있다. 각 항목이 선택된 순서를 순위로 생각하면 순위는 $\{1, \cdots, p\}$에 대한 순열 π로 나타낼 수 있다. 그리고 각 순위에 해당하는 항목의 인덱스는 순열 π의 역함수가 된다. 예를 들어, 집합 $S = \{1, 2, 3\}$의 원소들을 $(3 \to 1 \to 2)$ 순서로 선택한 경우, 각 항목의 순위는 순열 $\pi = (2, 3, 1)$이 되고 각 순위에 해당하는 항목 인덱스, 즉 각 항목이 선택된 순서는 순열의 역함수인 $\pi^{-1} = (3, 1, 2)$가 된다.

2.1. Estimation

Luce 모형의 모수를 추정하기 위해서는 우선 가능도함수 식을 정의할 필요가 있다. 선택 가능한 총 p개 항목의 집합을 S라 하고, S의 부분집합에 포함된 항목들에 대해 순위를 매기는 과정을 n번 시행했다고 가정한다. i번째 순위 결과는 부분집합 S_i에 포함된 항목들에 대해 순위를 매긴 것이다. S_i의 원소 개수를 p_i, i번째 순위 결과에 대해 선호도가 높은 순서대로 항목 인덱스를 나열한 벡터를 $\boldsymbol{r}_i = (r_{i1}, \cdots, r_{ip_i})$라 하겠다. Luce 모형의 모수 $\boldsymbol{u} = (u_1, \cdots, u_p)$에 대해 가능도함수는 아래와 같이 나타낼 수 있다.

$$L(\boldsymbol{u}) = \prod_{i=1}^{n} P(\boldsymbol{r}_i;\ \boldsymbol{u}) = \prod_{i=1}^{n} \prod_{j=1}^{p_i - 1} \frac{u_{r_{ij}}}{\sum_{k=j}^{p_i} u_{r_{ik}}} \tag{2.3}$$

여기서 모형의 식별성을 위해 $u_p = 1$이라는 제약조건을 둔다. 가능도함수를 최대화하는 최대 가능도 추정량으로 Luce 모형의 모수를 추정할 수 있다.

Luce 모형의 모수를 추정하는 또 다른 방법으로는 유사 가능도함수(pseudo likelihood function)를 활용하는 방법이 있다. 유사 가능도함수는 Luce 모형에서 항목을 쌍별 비교한 결과의 주변 확률을 이용한 것으로, 아래와 같이 주변 확률들의 곱으로 나타낼 수 있다. 여기서 y_{ijk}는 $I(r_{ij} < r_{ik})$, 즉 i번째 비교에서

j의 순위가 k의 순위보다 높은지 식별한 값이다.

$$L^{ps}(\boldsymbol{u}) = \prod_{i=1}^{n} \prod_{j,k \in S_i} \left(\frac{u_j}{u_j + u_k} \right)^{y_{ijk}} \tag{2.4}$$

유사 가능도함수에도 모형의 식별성을 위해 $u_p = 1$이라는 제약조건을 둔다. 일반적으로 모수 추정 시 유사 가능도함수를 이용하면 계산 과정이 더 간단해지거나 추정치에 대한 closed form이 존재하여 더 쉽게 구할 수 있다고 알려져 있다. 최대 유사 가능도함수 추정량은 Newton-Raphson 방법 등을 이용하여 구할 수 있다[6].

2.2. Regularization

정규화(regularization) 방법은 모수 추정 시 로그 가능도함수 또는 로그 유사 가능도함수에 음이 아닌 값을 갖는 penalty 함수를 적용하는 방법이다. penalty 함수는 모형의 complexity를 조절함으로써 모형의 성능을 향상시킨다. 순위 모형의 경우, 비슷한 선호도를 갖는 항목끼리 그룹으로 묶어주기 위한 목적으로 정규화 방법을 적용하는 경우가 많다. 실제로 이러한 정규화 방법이 순위 예측 시 test error를 상당히 감소시킨다는 것이 알려져 있다[9]. 이와 같은 결과를 얻기 위해서는 기존 모형에 항목 간 선호도 차이들의 총합이 일정 수준보다 작아지는 제약조건을 두면 된다. 예를 들어, Luce 모형의 로그 가능도함수에 항목 간 선호도 차이에 대한 penalty 함수 $G_\lambda(\cdot)$를 적용한 정규화 식은 아래와 같이 나타낼 수 있다.

$$-l(\boldsymbol{u}) + \sum_{j<k} G_\lambda(|u_j - u_k|) \tag{2.5}$$

위의 식에서 각 $\lambda \geq 0$에 대해 식을 최소화하는 u를 구할 수 있고, 이를 λ 값의 변화에 따라 solution path로 나타낼 수 있다. $\lambda = \infty$이면 penalty 함수의

효과가 매우 강력하여 모든 u_i가 같은 값을 갖게 된다. λ를 점점 감소시키면 선호도의 유사도가 상대적으로 떨어지는 항목부터 u_i 값이 분리되어 나오고, $\lambda = 0$이 되면 penalty 함수를 적용하지 않은 모형과 동일해지므로 모든 u_i가 다른 값을 갖게 된다. 이처럼 적절한 정규화 방법을 적용한 Luce 모형 식을 최소화함으로써 모수를 추정함과 동시에 유사한 선호도를 갖는 항목들을 그룹화할 수 있게 된다.

식 (2.5)에서 penalty 함수를 아래와 같이 각 쌍별 비교에 대한 가중치를 추가한 lasso-type penalty로 사용한 모형을 ranking lasso라 한다[9]. 일반적으로 ranking lasso는 쌍별 비교를 기반으로 한 Bradley-Terry 모형의 가능도함수를 이용한다. 아래 식에서 $l^{BD}(\boldsymbol{u})$는 Bradley-Terry 모형의 로그 가능도함수이다.

$$-l^{BD}(\boldsymbol{u}) + \lambda \sum_{j<k} w_{jk}|u_j - u_k| \tag{2.6}$$

ranking lasso에 이용되는 penalty 함수는 fused lasso penalty[12]를 일반화한 함수이다. fused lasso는 어떠한 자연적 순서를 갖고 있는 계수들을 축소하는 문제를 풀기 위해 만들어진 방법이기에 바로 인접한 계수들 간의 쌍별 차이에 대해서만 penalty를 부과하면 된다. 그러나 ranking lasso 문제에서는 항목들이 순서를 갖지 않으므로 모든 쌍별 비교를 고려한 penalty 함수를 이용한다.

식 (2.6)에서 쌍별 비교에 대한 가중치로 최대 가능도 추정치의 역수에 비례하는 아래의 가중치를 사용한 모형을 adaptive ranking lasso라 한다. 여기서 $\hat{u}^{ML}$은 최대 가능도 추정량을 말한다.

$$w_{jk} = |\hat{u}_j^{ML} - \hat{u}_k^{ML}|^{-1} \tag{2.7}$$

표본 크기가 커지면 쌍별 차이가 0이 아닌 경우에 대해서는 가중치가 유한한 상수로 수렴하게 되고 쌍별 차이가 0인 경우에 대해서는 해당 가중치가 발산하게 된다. 그러므로 (2.7)의 가중치를 사용하면 가중치를 모두 1로 동일하게

설정한 nonadaptive ranking lasso 모형보다 항목 간 선호도 차이를 다소 크게 추정하게 된다. 이로 인해 nonadaptive 모형 결과에서는 λ값 변화에 따라 solution path가 묶이는 과정에서 path가 서로 많이 겹쳐 잘 알아보기 힘든 반면, adaptive 모형 결과에서는 path가 완만한 형태로 나타나고 묶이는 지점도 보다 명확하게 분리되어 나타난다[9].

Bradley-Terry 모형을 활용한 ranking lasso는 쌍별 비교로 모형이 한정되어 있다는 단점이 있다. ranking lasso에서는 모형이 각 항목의 선호도의 로그값을 모수로 이용하고 있어 모수 추정 시 Newton-Raphson 방법을 이용하게 되는데, Newton-Raphson 방법에서는 Hessian matrix의 역행렬을 구해야 하므로 매우 많은 양의 계산이 필요하게 된다. 그리고 비교하는 항목의 수가 많아질수록 계산 결과 또한 불안정해진다는 문제점이 있다. 이러한 단점을 극복하기 위해 등장한 모형이 아래의 sparse Luce 모형이다[5].

$$-l(\boldsymbol{u}) + \lambda \sum_{j<k} |u_j - u_k| \tag{2.8}$$

sparse Luce 모형은 Luce 모형의 가능도함수에 바로 정규화 방법을 적용한 방법이다. 그러므로 기존의 ranking lasso와 달리 항목의 선호도 자체를 모수로 사용하게 되어 모수 추정 시 효율적인 계산 알고리즘을 활용할 수 있게 된다.

우리도 sparse Luce 모형과 같이 Luce 모형의 가능도함수에 바로 정규화 방법을 적용할 것이다. 그러나 지금까지 다룬 모형들처럼 항목 단위의 순위를 추정하는 것과는 달리, 우리는 비교할 항목이 여러 그룹으로 나누어져 있는 경우에 대해 그룹 단위의 순위 비교를 하고자 한다. NFL을 예로 생각해보면, NFL 팀들은 크게 두 개의 conference로 나누어져 있고, 각 conference도 다시 4개 지구로 나누어져 있다. 지금까지의 방법으로는 각 팀의 순위만을 추정할 수 있으므로 만일 어느 conference나 지구가 가장 우수한지, conference나 지구별 순위가 어떠한지는 알 수 없다. 이처럼 항목 단위의 순위 비교를 넘어 각

항목이 속한 집단들의 순위를 추정하기 위해 group penalty를 적용한 새로운 방법을 제안하고자 한다.

2.3. Group ranking lasso

2.2절에서 언급한 NFL 예제와 같이 항목들이 2개 이상의 그룹으로 나누어져 있는 경우, Luce 모형에 각 그룹에 대한 group penalty 함수를 적용함으로써 그룹별 기량을 비교할 수 있다. 순위 모형에 대한 기존의 정규화 방법들은 대체로 전체 모수에 대해 동일한 형태의 penalty 함수와 공통의 λ를 사용하지만, 우리는 그룹 내 비교와 그룹 간 비교에 대해 각각 서로 독립적인 penalty를 적용한다. 아래 Figure 2.1은 기존의 penalty 함수와 group penalty 함수의 penalty 형태를 그림으로 나타낸 것이다. 큰 정사각형의 가로와 세로에는 총 p개의 항목들이 각각 그룹 인덱스 순서대로 배치되어 있고, 작은 정사각형들은 이 항목들을 4개의 그룹으로 나눈 것을 의미한다. 왼쪽 그림에서는 group에 상관없이 쌍별 비교에 같은 penalty를 적용한 것을 나타내고 있다. 반면에 오른쪽 그림에서는 동일한 group에 속한 항목 간 쌍별 비교와 서로 다른 group에 속한 항목 간 쌍별 비교에 대해 서로 다른 penalty를 적용한 것을 나타내고 있다.

Luce 모형의 로그 가능도함수에 group penalty를 적용한 식은 아래와 같다. 모수 $\boldsymbol{u}$에 대해 각 항목의 그룹 인덱스를 적용한 이중인덱스를 부여하였다. 편의상 $\boldsymbol{u}$를 그룹 인덱스에 따라 정렬하여 기존 인덱스 번호 순서와 그룹 인덱스 순서와 동일하다고 가정한다. λ_1은 그룹 내 쌍별 비교에 관한 penalty를, λ_2는 그룹 간 쌍별 비교에 관한 penalty를 나타낸다.

$$-l(\boldsymbol{u}) + \lambda_1 \sum_{g} \sum_{j<k} |u_{g_j} - u_{g_k}| + \lambda_2 \sum_{g<h} \sum_{j,k} |u_{g_j} - u_{h_k}| \tag{2.9}$$

기존 penalty 함수와 마찬가지로 group penalty도 λ_1과 λ_2의 값을 변화시키며

그룹별 기량을 비교해볼 수 있다. 이제부터 식 (2.9)를 최소화하는 문제에 대해 생각하기로 한다.

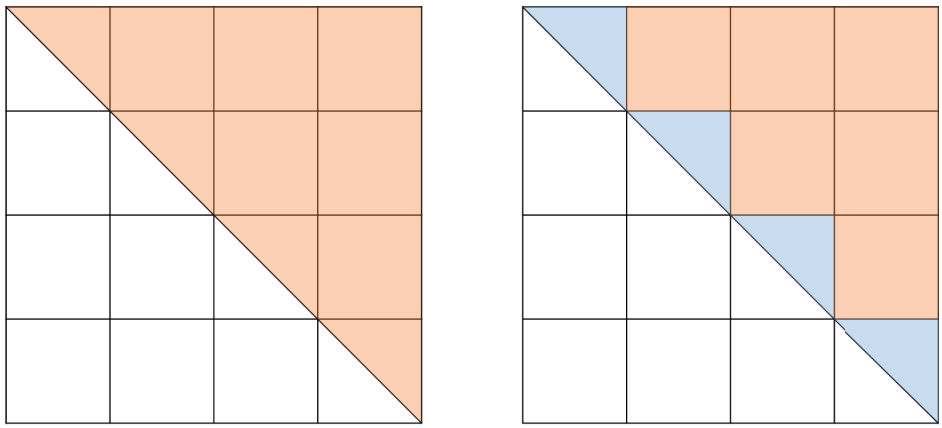

Figure 2.1: 기존의 lasso penalty 함수(왼쪽)와 group penalty 함수(오른쪽)

Chapter 3

Optimization

앞 장에서 언급한 바와 같이 우리가 풀고자 하는 문제는 Luce 모형에 group penalty를 적용한 문제로, 아래와 같이 나타낼 수 있다.

$$\min_{\boldsymbol{u}\geq 0, u_p=1} -l(\boldsymbol{u}) + \lambda_1 \sum_g \sum_{j<k} |u_{g_j} - u_{g_k}| + \lambda_2 \sum_{g<h} \sum_{j,k} |u_{g_j} - u_{h_k}|$$

기존의 문제에 대해 $\boldsymbol{\theta} = (\theta_{g_j h_k})$, $\theta_{g_j h_k} = u_{g_j} - u_{h_k}$로 정의하면 위 문제는 아래와 같이 바꿔쓸 수 있다.

$$\begin{aligned} &\min_{\boldsymbol{u}\geq 0, u_p=1, \boldsymbol{\theta}} -l(\boldsymbol{u}) + \lambda_1 \sum_g \sum_{j<k} |\theta_{g_j g_k}| + \lambda_2 \sum_{g<h} \sum_{j,k} |\theta_{g_j h_k}| \\ &\text{subject to } \theta_{g_j h_k} = u_{g_j} - u_{h_k} \end{aligned} \tag{3.1}$$

이 문제에 대해 ADMM 알고리즘을 적용하여 최적의 해를 찾도록 하겠다.

3.1. ADMM

ADMM(Alternating Direction Method of Multipliers) 알고리즘은 복잡한 최적화 문제를 간단한 하위 문제들로 분리함으로써 문제를 풀기 쉽게 변형하여

푸는 방법이다. ADMM 알고리즘 문제의 기본 형태는 다음과 같다.

$$\begin{aligned} &\text{minimize } f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ &\text{subject to } A\boldsymbol{x} + B\boldsymbol{z} = c \end{aligned} \tag{3.2}$$

여기서 f와 g는 볼록함수이고, 각각 $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{z} \in \mathbb{R}^m$, $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{p\times m}$, $c \in \mathbb{R}^p$라 가정한다. 기존의 augmented Lagrangian 방법이 변수 $\boldsymbol{x}$에 대한 제약조건 $h_i(\boldsymbol{x}) = 0$, $i = 1, \cdots, q$에 대해 $f(\boldsymbol{x})$를 최소화하는 문제였다면, ADMM 알고리즘에서는 변수가 $\boldsymbol{x}$와 $\boldsymbol{z}$로 분리되어 있고 목적함수 또한 변수별로 나뉜 것을 볼 수 있다. 그러나 식 (3.2)의 제약조건이 두 변수에 대해 분리되어 있지 않아 각 변수에 대해 독립적으로 문제를 푸는 것은 불가능하다. 대신 augmented Lagrangian 방법과 block descent를 이용해 문제를 쉽게 변형하여 풀 수 있다. 식 (3.2)에 대한 augmented Lagrangian 함수는 다음과 같다.

$$\mathcal{L}_\nu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\gamma}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\gamma}^{\boldsymbol{T}}(A\boldsymbol{x} + B\boldsymbol{z} - c) + \frac{\nu}{2}\|A\boldsymbol{x} + B\boldsymbol{z} - c\|_2^2$$

ADMM 알고리즘에서는 $\mathcal{L}_\nu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\gamma})$를 $\boldsymbol{x}$와 $\boldsymbol{z}$ 각각에 대해 block descent로 최소화한 후 $\boldsymbol{\gamma}$를 업데이트한다. 즉, 아래의 과정을 반복한다.

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &\leftarrow \underset{\boldsymbol{x}}{\operatorname{argmin}}\, \mathcal{L}_\nu(\boldsymbol{x}, \boldsymbol{z}^{(k)}, \boldsymbol{\gamma}^{(k)}) \\ \boldsymbol{z}^{(k+1)} &\leftarrow \underset{\boldsymbol{z}}{\operatorname{argmin}}\, \mathcal{L}_\nu(\boldsymbol{x}^{(k+1)}, \boldsymbol{z}, \boldsymbol{\gamma}^{(k)}) \\ \boldsymbol{\gamma}^{(k+1)} &\leftarrow \boldsymbol{\gamma}^{(k)} + \nu(A\boldsymbol{x}^{(k+1)} + B\boldsymbol{z}^{(k+1)} - c) \end{aligned}$$

기존의 augmented Lagrangian 방법에서 모든 변수에 대해 한 번 만에 함수를 최소화하는 것과는 달리, ADMM 알고리즘에서는 변수 각각에 대해 차례로 block descent를 적용함으로써 계산을 보다 간편하게 만들어준다[7].

이제 ADMM 알고리즘을 활용하여 우리의 문제를 풀도록 하겠다. 식 (3.1)

에 대한 augmented Lagrangian 함수는 다음과 같다.

$$\begin{aligned}\mathcal{L}_\nu(\boldsymbol{u},\boldsymbol{\theta},\boldsymbol{\gamma}) = &- l(\boldsymbol{u}) + \lambda_1 \sum_g \sum_{j<k} |\theta_{g_j g_k}| + \lambda_2 \sum_{g<h} \sum_{j,k} |\theta_{g_j h_k}| \\ &+ \sum_g \sum_{j<k} \Big[\gamma_{g_j g_k}(\theta_{g_j g_k} - u_{g_j} + u_{g_k}) + \frac{\nu}{2}(\theta_{g_j g_k} - u_{g_j} + u_{g_k})^2\Big] \\ &+ \sum_{g<h} \sum_{j,k} \Big[\gamma_{g_j h_k}(\theta_{g_j h_k} - u_{g_j} + u_{h_k}) + \frac{\nu}{2}(\theta_{g_j h_k} - u_{g_j} + u_{h_k})^2\Big]\end{aligned} \tag{3.3}$$

$\boldsymbol{\gamma} = (\gamma_{g_j h_k})$, $\gamma_{g_j h_k} \in \mathbb{R}$는 Lagrangian multiplier이고, ν는 양의 penalty에 관한 모수이다. ADMM 알고리즘은 식 (3.3)을 $\boldsymbol{u}$와 $\boldsymbol{\theta}$ 각각에 대해 식을 최소화한 후 새롭게 구한 $\boldsymbol{u}$와 $\boldsymbol{\theta}$를 바탕으로 $\boldsymbol{\gamma}$를 갱신하는 과정을 반복한다. 구체적인 알고리즘은 아래와 같다.

Algorithm 1. ADMM algorithm

1. 초깃값 $\boldsymbol{u}^{(0)}, \boldsymbol{\theta}^{(0)}, \boldsymbol{\gamma}^{(0)}$을 설정한다.
2. 아래의 과정을 $\mathcal{L}_\nu(\boldsymbol{u},\boldsymbol{\theta},\boldsymbol{\gamma})$이 수렴할 때까지 반복한다. $(t = 0, 1, \cdots)$
   (1) $\boldsymbol{u}^{(t+1)} = \text{argmin}_{\boldsymbol{u}}\, \mathcal{L}_\nu(\boldsymbol{u}, \boldsymbol{\theta}^{(t)}, \boldsymbol{\gamma}^{(t)})$
   (2) $\boldsymbol{\theta}^{(t+1)} = \text{argmin}_{\boldsymbol{\theta}}\, \mathcal{L}_\nu(\boldsymbol{u}^{(t+1)}, \boldsymbol{\theta}, \boldsymbol{\gamma}^{(t)})$
   (3) $\gamma_{g_j h_k}^{(t+1)} = \gamma_{g_j h_k}^{(t)} + \nu(\theta_{g_j h_k}^{(t+1)} - u_{g_j}^{(t+1)} + u_{h_k}^{(t+1)})$

Algorithm 1에서 볼 수 있듯이, ADMM 알고리즘은 문제를 3단계로 나누어 푼다. $\boldsymbol{\theta}$에 대해 최소화하는 (2-2) 과정은 soft thresholding rule에 의한 closed form solution이 존재하여 아래와 같이 쉽게 풀 수 있다.

$$\begin{aligned}\hat{\theta}_{g_j g_k} &= \text{sign}(\tilde{\theta}_{g_j g_k})\Big(|\tilde{\theta}_{g_j g_k}| - \frac{\lambda_1}{\nu}\Big)_+ \\ \hat{\theta}_{g_j h_k} &= \text{sign}(\tilde{\theta}_{g_j h_k})\Big(|\tilde{\theta}_{g_j h_k}| - \frac{\lambda_2}{\nu}\Big)_+ \quad (g \neq h)\end{aligned}$$

$\boldsymbol{u}$에 대한 최소화 문제 (2-1)은 $l(\boldsymbol{u})$가 두 번 미분 가능한 볼록함수임을 통해 Newton-Raphson 방법을 적용하여 최소화하는 $\boldsymbol{u}$를 구할 수 있다. 그러나 Newton-Raphson 방법은 비교하는 항목의 개수 p가 커지면 Hessian matrix와 그 역행렬을 계산하는데 많은 비용이 든다. 또한, Newton-Raphson 방법은 알고리즘의 수렴이 초깃값에 영향을 받으며 Hessian matrix가 거의 singular인 경우에는 계산 결과 또한 안정적이지 않다는 단점이 있다.

이러한 계산상의 문제를 피하기 위하여 제안된 방법이 cyclic MM 알고리즘이다[4]. 이 논문[4]은 Luce 모형의 최대 가능도 추정량을 구할 때 가능도함수에 대한 간단한 형태의 majorized function을 이용하는 cyclic MM 알고리즘을 활용하면 Newton-Raphson 방법보다 계산 시간이 단축된다는 것을 증명하였다. cyclic MM 알고리즘의 주된 장점은 Hessian matrix의 역행렬을 구할 필요가 없다는 것이다. 또한, MM 알고리즘 자체가 함숫값이 점차 감소하는 성질을 갖고 있다는 점도 우리의 문제를 푸는 데 있어 장점으로 작용한다. 이와 같은 사실을 바탕으로 우리는 (2-1) 과정에 cyclic MM을 적용하였다.

3.2. Cyclic MM

이번 절에서는 cyclic MM 알고리즘을 이용하여 주어진 $\boldsymbol{\theta}$, $\boldsymbol{\gamma}$에 대해 augmented Lagrangian 함수 $\mathcal{L}_\nu(\boldsymbol{u}, \boldsymbol{\theta}, \boldsymbol{\gamma})$를 최소화하는 $\boldsymbol{u}$를 구할 것이다. 이 절에서는 $\boldsymbol{\theta}$와 $\boldsymbol{\gamma}$는 상수처럼 생각하기로 한다. 일반적인 MM 알고리즘은 주어진 함수 $f(\boldsymbol{x})$에 대해 current solution $\boldsymbol{x}^{(n)}$에서의 majorized function $g(\boldsymbol{x}|\boldsymbol{x}^{(n)})$을 찾은 후, $g(\boldsymbol{x}|\boldsymbol{x}^{(n)})$를 최소화하는 해 $\boldsymbol{x}^{(n+1)}$로 업데이트하는 방법이다. 여기서 majorized function은 다음의 두 가지 조건을 만족해야한다.

$$\begin{aligned} f(\boldsymbol{x}^{(n)}) &= g(\boldsymbol{x}^{(n)}|\boldsymbol{x}^{(n)}) \\ f(\boldsymbol{x}) &\leq g(\boldsymbol{x}|\boldsymbol{x}^{(n)}),\ \forall \boldsymbol{x} \end{aligned} \tag{3.4}$$

위 조건을 만족하는 $g(\boldsymbol{x}|\boldsymbol{x}^{(n)})$에 대해 descent property가 성립하는 것을 아래와 같이 확인할 수 있다.

$$f(\boldsymbol{x}^{(n+1)}) \leq g(\boldsymbol{x}^{(n+1)}|\boldsymbol{x}^{(n)}) \leq g(\boldsymbol{x}^{(n)}|\boldsymbol{x}^{(n)}) = f(\boldsymbol{x}^{(n)})$$

cyclic MM 알고리즘은 MM 알고리즘의 특수 버전으로, 일반적인 MM 알고리즘과의 차이점은 cyclic MM 알고리즘의 경우 majorized function을 업데이트 할 때 1차원 방향으로 최적화한다는 것이다.

cyclic MM 알고리즘의 주요 아이디어는 $h_k \neq g_j$인 모든 h_k에 대해 $u_{h_k} = \tilde{u}_{h_k}$로 설정한 함수 $Q(\boldsymbol{u}|\tilde{\boldsymbol{u}})$가 closed form의 minimizer를 갖는 볼록함수라는 것이다. g_j번째 좌표에 대한 minimizer로 u_{g_j} 값이 업데이트 되면 (g_j+1)번째 좌표에 대해서도 동일한 방식으로 계산하여 다음 좌푯값 u_{g_j+1}을 업데이트하게 된다. 이처럼 $\boldsymbol{u}$의 첫 번째 좌표부터 마지막 좌표까지 순차적으로 1차원 최적화를 함으로써 $\boldsymbol{u}$에 대한 최적화 과정의 한 주기를 완성하게 된다. 이러한 주기를 반복하여 Luce 모형에 대한 최대 가능도 추정량을 구할 수 있다.

cylic MM 알고리즘의 첫 번째 단계로 $\mathcal{L}_\nu(\boldsymbol{u}, \boldsymbol{\theta}, \boldsymbol{\gamma})$에 대한 majorized function을 찾는다. 우선 로그 함수의 오목성에 의해 양수 x, y에 대하여 다음이 성립한다.

$$-\ln x \geq 1 - \ln y - (x/y), \quad \text{등호가 성립할 필요충분조건은 } x = y$$

위 성질을 이용하여 Luce 모형의 음의 로그 가능도함수에 대한 majorized function을 구할 수 있다. 알고리즘의 current solution을 $\tilde{\boldsymbol{u}}$라 하면 다음이 성립한다. 편의상 $\boldsymbol{u}$의 인덱스 번호는 식 (2.3)과 동일하게 사용한다.

$$-l(\boldsymbol{u}) = -\sum_{i=1}^{n}\sum_{j=1}^{p_i-1}\Big[\log u_{r_{ij}} - \log\Big(\sum_{k=j}^{p_i} u_{r_{ik}}\Big)\Big] \tag{3.5}$$

$$\leq -\sum_{i=1}^{n}\sum_{j=1}^{p_i-1}\Big[\log u_{r_{ij}} - \frac{\sum_{k=j}^{p_i} u_{r_{ik}}}{\sum_{k=j}^{p_i} \tilde{u}_{r_{ik}}} - \log\Big(\sum_{k=j}^{p_i} \tilde{u}_{r_{ik}}\Big) + 1\Big] \tag{3.6}$$

식 (3.6)을 $\tilde{l}(\boldsymbol{u}|\tilde{\boldsymbol{u}})$로 정의하면 $\tilde{l}(\boldsymbol{u}|\tilde{\boldsymbol{u}}) \geq -l(\boldsymbol{u})$이고 등호는 $\boldsymbol{u} = \tilde{\boldsymbol{u}}$일 때만 성립하므로 (3.4)의 두 조건을 만족한다. 즉, $\tilde{l}(\boldsymbol{u}|\tilde{\boldsymbol{u}})$는 current solution $\tilde{\boldsymbol{u}}$에서의 가능도함수에 대한 majorized function이 된다. 이를 이용하여 $\mathcal{L}_\nu(\boldsymbol{u}, \boldsymbol{\theta}, \boldsymbol{\gamma})$에 대한 majorized function $Q(\boldsymbol{u}|\tilde{\boldsymbol{u}})$를 식 (3.7)과 같이 구할 수 있다. 앞서 언급했듯이 현재 우리의 cyclic MM 알고리즘에서는 $\boldsymbol{\theta}$, $\boldsymbol{\gamma}$를 주어진 상수로 생각하고 있으므로, $\mathcal{L}_\nu(\boldsymbol{u}, \boldsymbol{\theta}, \boldsymbol{\gamma})$에서 $\boldsymbol{u}$와 무관한 부분인 $\lambda_1 \sum_g \sum_{j<k} |\theta_{g_j g_k}| + \lambda_2 \sum_{g<h} \sum_{j,k} |\theta_{g_j h_k}|$은 상수로 취급하여 majorized function을 구할 때 따로 고려하지 않아도 무방하다.

$$\begin{aligned} Q(\boldsymbol{u}|\tilde{\boldsymbol{u}}) :=& \tilde{l}(\boldsymbol{u}|\tilde{\boldsymbol{u}}) + \sum_g \sum_{j<k} \Big[\gamma_{g_j g_k}(\theta_{g_j g_k} - u_{g_j} + u_{g_k}) + \frac{\nu}{2}(\theta_{g_j g_k} - u_{g_j} + u_{g_k})^2\Big] \\ &+ \sum_{g<h} \sum_{j,k} \Big[\gamma_{g_j h_k}(\theta_{g_j h_k} - u_{g_j} + u_{h_k}) + \frac{\nu}{2}(\theta_{g_j h_k} - u_{g_j} + u_{h_k})^2\Big] \qquad (3.7) \\ &\geq \mathcal{L}_\nu(\boldsymbol{u}, \boldsymbol{\theta}, \boldsymbol{\gamma}) \text{ (등호는 } \boldsymbol{u} = \tilde{\boldsymbol{u}} \text{ 일 때만 성립)} \end{aligned}$$

다음으로, $Q(\boldsymbol{u}|\tilde{\boldsymbol{u}})$를 최소화하는 $\boldsymbol{u}$를 찾는다. cyclic MM 알고리즘에서는 각 좌표를 순차적으로 업데이트한다. 그러므로 현재 업데이트하는 g_j번째 좌푯값만을 변수로 생각하고 나머지 좌푯값은 current solution $\tilde{\boldsymbol{u}}$의 좌푯값으로 고정한 함수 $Q_{g_j}(\boldsymbol{u} = (\tilde{u}_1, \cdots, u_{g_j}, \cdots, \tilde{u}_p)|\tilde{\boldsymbol{u}})$를 최소화하는 문제로 바꿔 생각할 수 있다. $Q_{g_j}(\boldsymbol{u}|\tilde{\boldsymbol{u}})$가 미분 가능하고 볼록함수이므로 $Q_{g_j}(\boldsymbol{u}|\tilde{\boldsymbol{u}})$를 u_{g_j}에 대해 미분함으로써 minimizer를 구할 수 있다.

$$\begin{aligned} &-\sum_{i=1}^{n} I\Big(\sum_{l=1}^{p_i - 1} \delta_{il g_j} > 0\Big) \frac{1}{u_{g_j}} + \sum_{i=1}^{n} \sum_{l=1}^{p_i - 1} \delta_{il g_j} \frac{1}{\sum_{m=l}^{p_i} \tilde{u}_m} \\ &+ \sum_{k>j} [-\gamma_{g_j g_k} - \nu(\theta_{g_j g_k} - u_{g_j} + \tilde{u}_{g_k})] + \sum_{k<j} [\gamma_{g_k g_g} + \nu(\theta_{g_k g_j} - \tilde{u}_{g_k} + u_{g_j})] \\ &+ \sum_{h>g} \sum_{k} [-\gamma_{g_j h_k} - \nu(\theta_{g_j h_k} - u_{g_j} + \tilde{u}_{h_k})] + \sum_{h<g} \sum_{k} [\gamma_{h_k g_j} + \nu(\theta_{h_k g_j} - \tilde{u}_{h_k} + u_{g_j})] \end{aligned}$$

$$\delta_{il g_j} = \begin{cases} 1 & \text{if } g_j \in \{r_{il}, \cdots, r_{ip_i}\} \\ 0 & \text{otherwise} \end{cases}$$

위 식을 u_{g_j}에 대해 나타내면 아래와 같은 꼴로 표현된다.

$$a_{g_j} u_{g_j} + b_{g_j} - \frac{c_{g_j}}{u_{g_j}} = 0 \tag{3.8}$$

가능한 모든 쌍들의 인덱스 집합을 $\mathcal{N} = \{(j,k) : 1 \le j < k \le p\}$라고 하자. g_j번째 항목과 순위 비교 대상이 된 모든 항목에 관한 인덱스 집합을 $\mathcal{N}_{g_j} = \{h_k : (g_j, h_k) \in \mathcal{N} \text{ or } (h_k, g_j) \in \mathcal{N}\}$라 하면 $a_{g_j} = \nu|\mathcal{N}_{g_j}|/2 \ge 0$이 된다. b_{g_j}는 다음과 같이 정리할 수 있다.

$$\begin{aligned} b_{g_j} = &\sum_{i=1}^{n} \sum_{l=1}^{p_i - 1} \frac{\delta_{i l g_j}}{\sum_{m=l}^{p_i} \tilde{u}_m} + \Big(\sum_{k<j} \gamma_{g_k g_j} - \sum_{k>j} \gamma_{g_j g_k} + \sum_{h<g} \sum_{k} \gamma_{h_k g_j} - \sum_{h>g} \sum_{k} \gamma_{g_j h_k} \Big) \\ &+ \nu \Big(\sum_{k<j} \tilde{\theta}_{g_k g_j} - \sum_{k>j} \tilde{\theta}_{g_j g_k} + \sum_{h<g} \sum_{k} \theta_{h_k g_j} - \sum_{h>g} \sum_{k} \theta_{g_j h_k} \Big) - \nu \sum_{h_k \ne g_j} \tilde{u}_{h_k} \end{aligned}$$

$c_{g_j} = \sum_{i=1}^{n} I(\sum_{l=1}^{p_i - 1} \delta_{i l g_j} > 0)$는 순위 비교 결과 g_j번째 항목이 마지막 등수가 아니었던 총횟수를 의미한다. 모든 순위 비교 결과에서 마지막 등수를 받은 항목이 존재하지 않는다면 c_{g_j}는 모두 양수가 된다. 모든 c_{g_j}가 양수일 조건은 Luce 모형에서 최대 가능도 추정량이 존재할 조건과 동일한 것으로 알려져 있다[4].

$a_{g_j} > 0$, $c_{g_j} > 0$이면 이차방정식 꼴의 식 (3.8)을 최소화하는 유일한 양의 해가 존재하며, 다음과 같이 구할 수 있다.

$$\hat{u}_{g_j} = \frac{-b_{g_j} + \sqrt{b_{g_j}^2 + 4 a_{g_j} c_{g_j}}}{2 a_{g_j}} \tag{3.9}$$

cyclic MM 알고리즘 과정을 요약하면 아래와 같다.

Algorithm 2. Cyclic MM algorithm

1. 초깃값 $\tilde{\boldsymbol{u}}^{(0)}$를 설정한다.
2. 아래의 과정을 $\tilde{\boldsymbol{u}}$의 각 좌표에 대해 순서대로 시행한다.
   (1) a_{g_j}, b_{g_j}, c_{g_j}를 계산한다.
   (2) 식 (3.9)에 따라 $\tilde{\boldsymbol{u}}$의 g_j번째 좌표값을 $\hat{u}_{g_j}$로 업데이트한다.
3. $\tilde{\boldsymbol{u}}$를 마지막 좌표값(편의상 $\tilde{u}_p$)를 이용해 다음과 같이 normalize한다.

$$\tilde{\boldsymbol{u}} \leftarrow \tilde{\boldsymbol{u}} \,/\, \tilde{u}_p$$

4. $\tilde{\boldsymbol{u}}$가 수렴할 때까지 2단계, 3단계를 반복한다.

cyclic MM 알고리즘의 세 번째 단계는 각 모수를 normalize 하는 과정으로, MM 알고리즘의 감소하는 성질을 위반할 수 있다. cyclic MM 알고리즘의 감소성을 만족하기 위해서는 $\tilde{\boldsymbol{u}}$의 마지막 좌표 $\tilde{u}_p$의 업데이트 과정을 변형하면 된다. 마지막 좌표 업데이트 과정에서 $\tilde{u}_p$를 업데이트하는 대신 $\tilde{u}_p$는 항상 1로 고정하고 나머지 좌표에 대한 scaling factor를 α로 둔다. 변수 α에 대해 $\mathcal{L}_\nu((\alpha u_1, \cdots, \alpha u_{p-1}, 1), \boldsymbol{\theta}, \boldsymbol{\gamma})$를 최소화하는 $\hat{\alpha}$를 구해 좌푯값들을 scaling 한다. 이렇게 scaling 하는 방법 역시 1차원 최적화 과정으로 closed form solution이 존재하지만, 위의 Algorithm 2에서 언급한 것과 같은 간단한 방법을 이용하여도 실제 알고리즘이 문제없이 작동한다는 것이 알려져있다[5]. 우리도 계산상의 편의를 위해 간단한 방법을 이용하였다.

3.3. Stopping Criterion

ADMM 문제 (3.2)의 최적해를 $(\boldsymbol{x}^*, \boldsymbol{z}^*, \boldsymbol{\gamma}^*)$라 하면, ADMM 알고리즘의 최적성에 대한 필요충분조건은 primal feasibility (3.10)과 dual feasibility (3.11),

(3.12)이다. 식 (3.11), (3.12)에서 ∂는 subdifferential 기호를 나타낸다.[3.1]

$$A\boldsymbol{x}^* + B\boldsymbol{z}^* - c = 0 \tag{3.10}$$

$$0 \in \partial f(\boldsymbol{x}^*) + A^T\boldsymbol{\gamma}^* \tag{3.11}$$

$$0 \in \partial g(\boldsymbol{z}^*) + B^T\boldsymbol{\gamma}^* \tag{3.12}$$

$\boldsymbol{z}^{(k+1)}$이 $\mathcal{L}_\nu(\boldsymbol{x}^{(k+1)}, \boldsymbol{z}, \boldsymbol{\gamma}^{(k)})$를 $\boldsymbol{z}$에 대해 최소화하는 값이므로 다음이 성립한다.

$$\begin{aligned} 0 &\in \partial g(\boldsymbol{z}^{(k+1)}) + B^T\boldsymbol{\gamma}^{(k)} + \nu B^T(A\boldsymbol{x}^{(k+1)} + B\boldsymbol{z}^{(k+1)} - c) \\ &= \partial g(\boldsymbol{z}^{(k+1)}) + B^T\boldsymbol{\gamma}^{(k+1)} \end{aligned} \tag{3.13}$$

위 결과를 통해 $\boldsymbol{z}^{(k+1)}$과 $\boldsymbol{\gamma}^{(k+1)}$이 항상 (3.12)을 만족하는 것을 알 수 있다. 그러므로 ADMM 알고리즘의 최적성은 (3.10), (3.11)의 성립 여부에 따라 결정된다. (3.13)에서 $\boldsymbol{r}^{(k+1)} := A\boldsymbol{x}^{(k+1)} + B\boldsymbol{z}^{(k+1)} - c$라 하면 $\boldsymbol{r}^{(k+1)}$는 primal feasibility 조건 (3.10)에 대한 residual로 생각할 수 있다.

마찬가지로, $\boldsymbol{x}^{(k+1)}$이 $\mathcal{L}_\nu(\boldsymbol{x}, \boldsymbol{z}^{(k)}, \boldsymbol{\gamma}^{(k)})$를 $\boldsymbol{x}$에 대해 최소화하는 값이므로 다음이 성립한다.

$$\begin{aligned} &0 \in \partial f(\boldsymbol{x}^{(k+1)}) + A^T\boldsymbol{\gamma}^{(k)} + \nu A^T(A\boldsymbol{x}^{(k+1)} + B\boldsymbol{z}^{(k)} - c) \\ &= \partial f(\boldsymbol{x}^{(k+1)}) + A^T\boldsymbol{\gamma}^{(k)} + \nu A^T\{(A\boldsymbol{x}^{(k+1)} + B\boldsymbol{z}^{(k+1)} - c) + B(\boldsymbol{z}^{(k)} - \boldsymbol{z}^{(k+1)})\} \\ &= \partial f(\boldsymbol{x}^{(k+1)}) + A^T\boldsymbol{\gamma}^{(k+1)} + \nu A^T B(\boldsymbol{z}^{(k)} - \boldsymbol{z}^{(k+1)}) \end{aligned}$$

위 결과는 아래 식으로 바꿔쓸 수 있다.

$$\nu A^T B(\boldsymbol{z}^{(k+1)} - \boldsymbol{z}^{(k)}) \in \partial f(\boldsymbol{x}^{(k+1)}) + A^T\boldsymbol{\gamma}^{(k+1)} \tag{3.14}$$

위 식에서 $\boldsymbol{s}^{(k+1)} := \nu A^T B(\boldsymbol{z}^{(k+1)} - \boldsymbol{z}^{(k)})$라 하면 $\boldsymbol{s}^{(k+1)}$는 dual feasibility 조건 (3.11)에 대한 residual로 생각할 수 있다. 이러한 특성을 바탕으로 $\boldsymbol{r}^{(k+1)}$과

[3.1] f, g가 모두 미분 가능한 경우에는 ∂f, ∂g, $\in$ 대신 ∇f, ∇g, $=$ 기호를 사용할 수 있다[2].

$\boldsymbol{s}^{(k+1)}$을 각각 $(k+1)$번째 반복에서의 primal residual, dual residual이라 한다. ADMM 알고리즘을 반복할수록 두 residual은 0으로 수렴한다.

최적성 조건에 대한 두 residual은 현재 지점에서의 준최적성(suboptimality)과 관련이 있다. 목적함수의 최적값을 p^*라 하면 다음의 부등식이 성립하는 것이 알려져 있다[2].

$$f(\boldsymbol{x}^{(k)}) + g(\boldsymbol{z}^{(k)}) - p^* \leq -(\boldsymbol{\gamma}^{(k)})^T \boldsymbol{r}^{(k)} + (\boldsymbol{x}^{(k)} - \boldsymbol{x}^*)^T \boldsymbol{s}^{(k)} \tag{3.15}$$

이는 $\boldsymbol{r}^{(k)}$와 $\boldsymbol{s}^{(k)}$가 작아지면 현재 지점에서의 목적함수의 함숫값과 최적값의 차이가 작아지게 된다는 것을 의미한다. 그러나 우리는 $\boldsymbol{x}^*$를 모르기 때문에 이 부등식을 stopping criterion으로 바로 적용하지는 못한다. 대신 현재의 $\boldsymbol{x}^{(k)}$와 $\boldsymbol{x}^*$의 l_2-norm 차이가 $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|_2 \leq d$로 유계되어 있다고 가정하면 아래의 부등식을 얻을 수 있고, 이를 현재 지점의 목적함수 값과 최적값의 차이에 대한 대략적인 상한값으로 사용할 수 있게 된다.

$$f(\boldsymbol{x}^{(k)}) + g(\boldsymbol{z}^{(k)}) - p^* \leq -(\boldsymbol{\gamma}^{(k)})^T \boldsymbol{r}^{(k)} + d\|\boldsymbol{s}^{(k)}\|_2 \leq \|\boldsymbol{\gamma}^{(k)}\|_2 \|\boldsymbol{r}^{(k)}\|_2 + d\|\boldsymbol{s}^{(k)}\|_2$$

이를 통해 $\|\boldsymbol{r}^{(k)}\|_2$, $\|\boldsymbol{s}^{(k)}\|_2$가 일정 수준 이하로 작아지면 목적함수의 함숫값이 최적값으로 수렴했다고 판단할 수 있다. 각 residual에 대한 구체적인 허용한계 기준은 다음과 같다.

$$\begin{aligned} \|\boldsymbol{r}^{(k)}\|_2 &\leq \epsilon_{pri} := \sqrt{p}\,\epsilon_{abs} + \epsilon_{rel} \max\{\|A\boldsymbol{x}^{(k)}\|_2,\ \|B\boldsymbol{z}^{(k)}\|_2,\ \|c\|_2\} \\ \|\boldsymbol{s}^{(k)}\|_2 &\leq \epsilon_{dual} := \sqrt{n}\,\epsilon_{abs} + \epsilon_{rel}\|A^T\boldsymbol{\gamma}^{(k)}\|_2 \end{aligned} \tag{3.16}$$

여기서 ϵ_{abs}은 absolute tolerance, ϵ_{rel}은 relative tolerance이며 모두 양수이다. ϵ_{rel}은 10^{-3}이나 10^{-4}를 주로 이용하고, ϵ_{abs}는 변수의 값 크기에 따라 설정한다. $\sqrt{p}$, $\sqrt{n}$은 각각의 l_2-norm이 $\mathbb{R}^p$, $\mathbb{R}^n$에서 계산된 것을 반영한 것이다[2].

그런데 이러한 이론적인 stopping criterion 대신, 현재 지점과 이전 반복에서의 변수 추정값의 변화가 작아지면 알고리즘을 중단하도록 하는 간단한

기준을 이용하여도 실제 알고리즘이 잘 작동하였다. 또한, 간단한 기준을 사용할 경우 매 반복마다 $\|\boldsymbol{r}^{(k)}\|_2$, $\|\boldsymbol{s}^{(k)}\|_2$를 계산하는 과정을 생략하므로 계산량도 적어져 계산 시간이 적게 걸렸다. 우리는 실제 데이터 분석 시 계산상의 편의를 위해 간단한 기준을 바탕으로 한 알고리즘을 이용하였다.

Chapter 4

Numerical analysis

제안한 방법을 적용한 결과를 알아보기 위해 2015년도 미국 프로 미식축구 리그(NFL) 정규 시즌 데이터를 이용하였다. 미국 프로 미식축구 리그에는 32개 팀이 있고 American Football Conference(AFC)와 National Football Conference(NFC)의 두 conference로 나누어져 있다. 두 conference는 각각 네 개의 지구로 나누어져 있으며, 지구별로 네 개의 팀이 속해있다. 정규 시즌에서는 팀마다 16번의 경기를 하는데, 그중 6번은 동일한 지구 내의 팀들과, 6번은 동일한 conference 내 다른 지구 팀들과, 그리고 나머지 4번은 다른 conference의 팀들과 경기를 한다. 2015년도 NFL 정규 시즌은 2015년 9월 10일부터 2016년 1월 3일까지 총 256번의 경기가 이루어졌다. 정규 시즌에서 비기는 경기가 있을 수 있으나, 연장전 제도가 생긴 이후로는 거의 없었으며 2015년도 NFL 정규 시즌에서도 비긴 경기는 없었다. 데이터는 `http://www.footballdb.com` 사이트를 참조하였다. 아래 표는 NFL 32개 팀에 대해 conference별, 지구별 분포를 나타낸 것이다. 각 conference의 지구마다 팀이 네 개씩 분포된 것을 확인할 수 있다. 우리는 이와 같이 총 8개 그룹으로 팀들이 나누어져 있다고 생각하고 제안한 모형을 적용하여 그룹별 기량을 비교하였다.

	American Football Conference	National Football Conference
East	Buffalo Miami New England NY Jets	Dallas NY Giants Philadelphia Wasington
North	Baltimore Cincinnati Cleveland Pittsburgh	Chicago Detroit Green Bay Minnesota
South	Houston Indianapolis Jacksonville Tennessee	Atlanta Carolina New Orleans Tampa Bay
West	Denver Kansas City Oakland San Diego	Arizona San Francisco Seattle St.Louis

Table 4.1: NFL 경기 팀들의 각 conference별, 지구별 팀 분포

우선 그룹 단위의 비교에 앞서, 팀별 기량을 비교해보았다. 여기서 '기량'이란 앞에서 Luce 모형에 대한 설명에서 사용했던 '선호도'와 같은 개념으로 생각하면 된다. 팀별 비교는 group penalty를 사용한 식 (3.1)에서 λ_1과 λ_2를 공통의 λ로 생각하고 알고리즘을 동일하게 적용하면 된다. 아래 Figure 4.1은 각 팀의 기량을 비교한 결과이다. Figure 4.1의 y축인 $\log(u)$는 각 팀의 기량을

수치로 나타낸 것으로, 기량의 로그값을 나타내고 있다. u가 클수록 기량이 뛰어난 것을 의미한다. 아래 결과를 통해 λ가 커질수록 팀 간 기량 차이에 대한 penalty가 점점 증가하여 기량의 차이가 점차 작아지고, 결국 한 개의 동일한 값(여기서는 0에 해당)으로 수렴하는 것을 볼 수 있다. 또한, 수렴하는 과정에서 비슷한 기량의 팀들끼리는 0으로 최종 수렴하기 이전에 먼저 동일한 값으로 묶이는 것도 확인할 수 있다.

이제 그룹 단위의 기량을 비교하기 위해서는 식 (3.1)에서 그룹 내 penalty λ_1을 일정 수준 이상으로 설정하는 방법을 생각할 수 있다. 동일한 그룹에 포함된 팀들 간의 기량 차이에 대해 일정 수준 이상의 큰 penalty를 부과하면 그룹 내 팀들 간의 기량 차이를 0에 근사한 값으로 축소 추정하게 된다. 즉, 각 그룹 내 팀들의 기량이 한 개의 값으로 묶여서 나타나게 된다. 그러므로 각 그룹마다 기량에 관한 한 개의 대푯값을 구할 수 있게 되므로, 이 값을 이용해 일변량의 단순 수치 비교가 가능해진다. 아래 Figure 4.2는 $\lambda_1 = 5$일 때 λ_2에 관한 solution path를 나타낸 것이다. Figure 4.1과는 달리 그룹마다 한 개의 path로 묶여서 나타나는 것을 확인할 수 있다.

Figure 4.2를 통해 NFL 2015 정규 시즌에서는 NFC North → NFC West → AFC North → NFC South → AFC Eest → AFC West → NFC East → AFC South 순서로 기량이 뛰어났던 것을 확인할 수 있다. 이처럼 그룹 내 penalty를 부여함으로써 그룹 단위의 비교를 할 수 있다.

실제로 NFL 2015 정규 시즌 경기에 대해 지구별 순위를 평가한, 다양한 의견들이 있었다. 한 가지 예로는, NFC West → AFC North → NFC North → AFC East → NFC East → AFC West → NFC South → AFC South 의 순위가 있다. 이처럼 일반적으로 사람들이 생각하는 순위와 우리의 모형 결과를 비교하면 상위권, 최하위권의 팀들은 대체로 비슷하였으나 그들의 정확한 순위는 다소 차이가 있었다. 이러한 현상은 우리 모형에서는 NFL 팀들에

대한 사전 정보 등이 고려되지 않았던 점이 원인일 수 있다. 실제로 사람들의 순위 평가 결과가 꽤 다양했던 것을 통해 순위 평가에 주관적인 의견이 많이 개입되었을 것이라 생각한다.

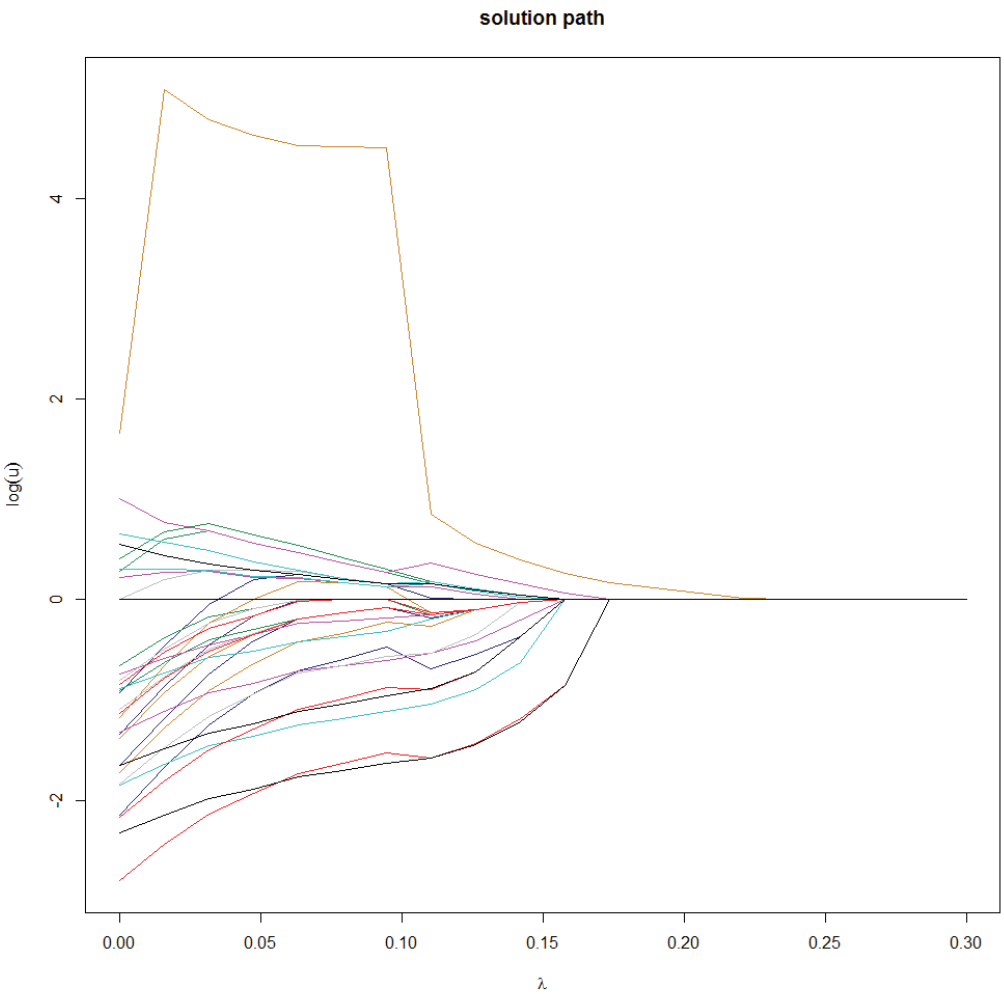

Figure 4.1: NFL 2015 데이터에 대한 solution path. 각 선은 각 팀의 기량을 나타내며, 선의 색은 그룹마다 다르게 나타내었다.

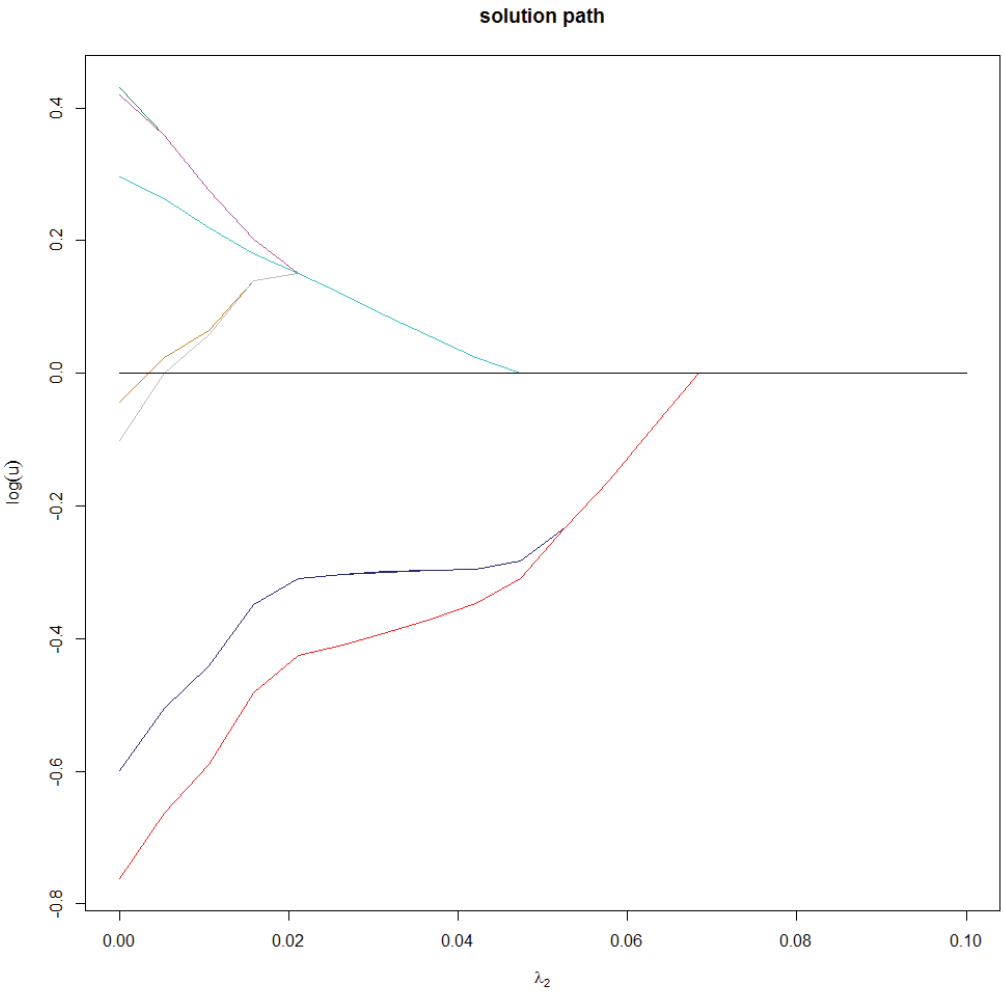

Figure 4.2: NFL 2015 데이터에 $\lambda_1 = 5$로 설정했을 때 λ_2에 관한 solution path. 각 선은 각 팀의 기량을 나타내며, 선의 색은 그룹마다 다르게 나타내었다.

Chapter 5

결론 및 제언

Luce 모형에 적절한 정규화 방법을 적용함으로써 항목별 순위뿐만 아니라 각 그룹의 순위도 추정할 수 있었다. 양의 모수들 간 차이에 대해 penalty를 적용함으로써 Luce 모형의 모수를 효율적으로 계산하였다. 특히 항목들이 여러 개의 그룹으로 나누어져 있는 경우, 전체 항목에 대해 동일하게 적용되던 기존의 penalty 함수를 그룹 내 penalty와 그룹 간 penalty로 분리하여 그룹 단위의 비교를 가능하게 하였다. 이때 그룹 내 penalty를 일정 수준 이상으로 설정하면 그룹 단위 비교가 용이해진다. 계산 알고리즘으로는 ADMM 알고리즘을 이용하였다. ADMM 알고리즘의 내부 과정에서 Luce 모형의 모수에 관한 추정치를 업데이트할 때 cyclic MM 알고리즘을 활용함으로써 기존의 Newton-Raphson 방법보다 계산 시간을 단축시키고 계산 결과의 안정성도 높일 수 있었다. 제안한 모형을 NFL 2015 정규 시즌 경기 데이터에 적용한 결과, 우선 기존의 penalty 함수를 적용했을 때에는 penalty 값의 변화에 따라 경기 승패 기록이 비슷한 팀끼리 같은 순위로 잘 묶이는 것을 확인할 수 있었다. group penalty 함수를 적용하였을 때는 그룹마다 팀들의 기량이 하나의 값으로 묶여서 나타났다. 기존 penalty 함수의 결과와 마찬가지로 그룹 간 penalty가

증가함에 따라 그룹 간 기량의 차이가 점차 줄어들고 비슷한 기량의 그룹끼리 묶이는 것도 확인할 수 있었다. 이처럼 기량을 나타내는 모수에 대한 추정치를 통해 각 팀의 순위와 그룹의 순위를 쉽게 파악할 수 있었다.

우리의 모형에서는 penalty 함수로 가장 간단한 형태인 lasso type penalty를 사용하였는데, 이 외에도 sparse한 형태로 축소추정하는 다양한 penalty 함수를 사용할 수 있을 것이다. 예를 들어, capped l_1-norm penalty 함수를 사용하면 기존의 lasso penalty를 사용했을 때보다 penalty의 변화에 따라 항목들이 같은 값으로 묶이는 것을 더 명확하게 확인할 수 있다[5]. 그리고 group penalty의 경우에는 일반적인 group lasso와 유사하게 l_2-norm penalty를 적용하는 방법도 생각해볼 수 있다. 예를 들어, 식 (3.1)에서 그룹 간 penalty 함수를 l_2-norm penalty로 두면 아래의 문제를 풀게 된다.

$$\min_{\boldsymbol{u}\geq 0, u_p=1, \boldsymbol{\theta}} -l(\boldsymbol{u}) + \lambda_1 \sum_{g}\sum_{j<k} |\theta_{g_j g_k}| + \lambda_2 \sum_{g<h} \sqrt{\sum_{j,k} \theta^2_{g_j h_k}}$$

위 식을 이용하면 서로 다른 그룹 간 비교에 대한 $\theta_{g_j h_k}$의 추정량이 기존의 방법과 달라지게 된다. 즉, 위 식에 대한 augmented Lagrangian 함수 중 아래의 항들을 최소화하는 값으로 $\theta_{g_j h_k}$를 추정하게 된다.

$$\lambda_2 \sum_{g<h} \sqrt{\sum_{j,k} \theta^2_{g_j h_k}} + + \sum_{g<h}\sum_{j,k} \left[\gamma_{g_j h_k} (\theta_{g_j h_k} - u_{g_j} + u_{h_k}) + \frac{\nu}{2} (\theta_{g_j h_k} - u_{g_j} + u_{h_k})^2 \right]$$

본론에서 $\boldsymbol{\theta}$를 추정할 때와 마찬가지로 위 식을 각각의 $\theta_{g_j h_k}$에 대해 미분한 후 미분한 식의 값이 0이 되도록 하는 $\hat{\theta}_{g_j h_k}$를 구할 수 있다. 위 식을 $\theta_{g_j h_k}$에 대해 미분하게 되면 $\theta_{g_j h_k}$에 대한 4차 방정식이 되므로 4차 방정식의 근의 공식을 활용해 $\hat{\theta}_{g_j h_k}$를 구할 수 있다. 또는, 행렬을 이용해 나타낸 아래 식을 활용할 수 있다.

$$\lambda_2 \sum_{g<h} \|\boldsymbol{\theta}_{gh}\|_2 + \sum_{g<h} \left[\boldsymbol{\gamma}_{gh}^T (\boldsymbol{\theta}_{gh} - D_{gh}\boldsymbol{u}) + \frac{\nu}{2} \|\boldsymbol{\theta}_{gh} - D_{gh}\boldsymbol{u}\|_2^2 \right]$$

여기서 $\boldsymbol{\theta}_{gh} = (\theta_{g_j h_k})$, $\boldsymbol{\gamma}_{gh} = (\gamma_{g_j h_k})$인 벡터이고, D_{gh}는 $\boldsymbol{u}$에서 g번째 그룹과 h번째 그룹의 쌍별 비교에 관한 행렬이다. 즉, 각각의 그룹 인덱스 g, h에 대해 $\boldsymbol{\theta_{gh}} - D_{gh}\boldsymbol{u} = 0$라는 제약조건이 주어진 상태이다. 마찬가지로 위의 행렬식을 $\boldsymbol{\theta}_{gh}$에 대해 미분한 후 미분한 식의 값이 0이 되도록 하는 추정량을 구한다.

$$\lambda_2 \frac{\boldsymbol{\theta}_{gh}}{\|\boldsymbol{\theta}_{gh}\|_2} + \boldsymbol{\gamma}_{gh} + \nu(\boldsymbol{\theta}_{gh} - D_{gh}\boldsymbol{u}) = 0$$

이처럼 다양한 penalty 함수를 적용하여 주어진 문제를 풀 수 있다. 본문에서 사용했던 penalty 함수와 다른 함수를 사용하더라도 계산 과정에서는 ADMM 알고리즘과 cyclic MM 알고리즘을 활용할 수 있다.

우리가 사용한 cyclic MM 알고리즘은 Luce 모형 외에도 고차원 범주형 데이터에 관한 로지스틱 정규화 회귀 모형 등의 다른 모형에도 적용할 수 있다[1]. 즉, 이 알고리즘은 순위 추정이라는 특정 문제에 한정된 알고리즘이 아니라 다양한 문제에 적용하여 계산을 효율적으로 할 수 있도록 만들어줄 것이다.

우리의 모형은 공동 순위가 없는 경우에만 적용할 수 있다는 한계점이 있다. 그러나 실제 데이터 중에는 동점을 얻거나 비기는 경우가 많다. 예를 들어, 스포츠 경기 중에도 연장전 제도가 없는 경기에서는 동점이 나올 수 있고, 영화 평점 데이터의 경우에도 평점의 등급이 매우 세분화되어 있지는 않으므로 한 명의 관람객에 대해 동일한 평점을 받은 영화가 많을 것이다. 이러한 경우까지 모두 다루기 위해서는 모형을 확장할 필요가 있다.

Chapter 6

부록

```
########## DATA SETTING ##########

# NFL DATA
nfldata <- read.csv('nfl2015.csv', header=TRUE)
rownames(nfldata) <- nfldata[, 1] ## set rownames
nfldata <- nfldata[, -1]
count_data <- as.matrix(nfldata)
n <- dim(count_data)[1]

# NFL GROUP
nflgroup <- read.csv('nfl2010_group_.csv', header=TRUE)
rownames(nflgroup) <- nflgroup[, 1] ## set rownames
nflgroup <- nflgroup[order(nflgroup$X),]
nflgroup <- nflgroup[, -1]
nfl <- cbind(count_data, nflgroup)
temp <- cbind(t(nfl[order(nfl$group,nfl$num),][,1:32]), nflgroup)
```

```
nflsort <- cbind(t(temp[order(nfl$group,nfl$num),][1:32]),
                 nflgroup[order(nflgroup$group,nflgroup$num),])
count <- nflsort[,1:32]

########## ALGORITHM ##########

# SOFT-THRESHOLDING
sft <- function(x, y) {
  sign(x) * max(abs(x) - y,0)
}

# D-matrix
D <- matrix(c(rep(1, (n-1)), diag(-1, (n-1))), nrow = (n-1))
for (i in 2:(n-1)) {
  D <- rbind(D, matrix(c(rep(0, (n-i) * (i-1)),
             rep(1, (n-i)), diag(-1, (n-i))), nrow = n-i))
}

lambda1 <- 5
lambda2 <- c(seq(0, 0.1, length.out = 20))
nu <- 1
maxiter <- 10000
grps <- 8
itms <- n / grps
W <- as.matrix(rowSums(count))
N <- as.matrix(count + t(count))
ufit <- matrix(nrow = n, ncol = length(lambda2))
```

```
for (l in 1:length(lambda2)) {
  iter <- 0
  u_prev <- matrix(1, nrow = n)
  theta_prev <- matrix(0, nrow = n, ncol = n)
  gamma <- matrix(0, nrow = n, ncol = n)

  while(TRUE) {
    iter <- iter + 1

    ## UPDATE U-VECTOR (CYCLIC MM)
    mmiter <- 0
    u_temp <- u_prev

    while (TRUE) {
      mmiter <- mmiter + 1
      u <- matrix(1, nrow = n)
      wt <- matrix(0, nrow = n)

      for (m in 1:n) {
        if (m == 1) {
          for (k in 1:n) {
            wt[k] <- 1 / (u_temp[m] + u_temp[k])
          }
          a <- (N[m,] %*% wt + nu * (- sum(theta_prev[m,2:n]))
               + (- sum(gamma[m,2:n])) - nu * sum(u_temp[2:n]))
        } else if (m == n) {
          for (k in 1:(m-1)) {
```

```
      wt[k] <- 1 / (u_temp[m] + u[k])
    }
    a <- (N[m,] %*% wt + nu * (sum(theta_prev[1:(m-1), m]))
        + (sum(gamma[1:(m-1), m])) - nu * sum(u[1:(m-1)]))
  } else {
    for (k in 1:(m-1)) {
      wt[k] <- 1 / (u_temp[m] + u[k])
    }
    for (k in m:n) {
      wt[k] <- 1 / (u_temp[m] + u_temp[k])
    }
    a <- (N[m,] %*% wt + nu * (sum(theta_prev[1:(m-1), m])
        - sum(theta_prev[m, (m+1):n]))
        + (sum(gamma[1:(m-1), m]) - sum(gamma[m, (m+1):n]))
        - nu * (sum(u[1:(m-1)]) + sum(u_temp[(m+1):n])))
  }
  u[m] <- (- a + sqrt(a^2 + 4*nu*(n-1)*W[m])) / (2*nu*(n-1))
}
u <- u / u[n]

unorm1 <- norm(u, 'E')
unorm2 <- norm(u_temp, 'E')
if (abs(unorm1-unorm2)/unorm1 < 10^-6 | mmiter == 10000) {
  break
}
u_temp <- u
}
```

```
## UPDATE THETA (SOFT THRESHOLDING)
theta <- matrix(0, nrow = n, ncol = n)

for (g in 0:(grps - 1)) {
for (h in g:(grps - 1)) {

  # within group
  if(g == h) {
    for (j in 1:itms) {
    for (k in j:itms) {

      if(j < k) {
        theta[g*itms+j,g*itms+k] <- sft(u[g*itms+j]-u[g*itms+k]
        - gamma[g*itms+j, g*itms+k] / nu, lambda1[1] / nu)
      }
    }
    }
  } else {  # between group
    for (j in 1:itms) {
    for (k in 1:itms) {
      theta[g*itms+j,h*itms+k] <- sft(u[g*itms+j]-u[h*itms+k]
      - gamma[g*itms+j, h*itms+k] / nu, lambda2[l] / nu)
    }
    }
  }
  }
}
```

```
    ## UPDATE GAMMA
    for (i in 1:n) {
      for (j in 1:n) {
        gamma[i,j] <- gamma[i,j] + nu * (theta[i,j] - u[i] + u[j])
      }
    }

    ## STOP
    if (max(abs(u - u_prev)) < 10^-6 | iter == maxiter) {
      break
    }

    u_prev <- u
    theta_prev <- theta
  }
  ufit[, l] <- u
}

cols = rep(c('navy', 'seagreen', 'orange3', 'magenta', 'gray',
             'cyan3', 'red', 'black'), each = 4)
plot(0, 0, type = 'n',
     xlim = c(0,lambda2[length(lambda2)]), ylim = range(log(ufit)),
     xlab = expression(lambda), ylab = '', main = 'solution path')
matplot(lambda2, t(log(ufit)),col=cols,type="l",lty=1,add=TRUE)
```

Bibliography

[1] Alan Agresti and Maria Kateri. *Categorical data analysis*. Springer, 2011.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[3] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[4] David R Hunter. Mm algorithms for generalized bradley-terry models. *Annals of Statistics*, pages 384–406, 2004.

[5] Jong-June Jeon and Hosik Choi. The sparse luce model. In *Applied Intelligence*, 2016.

[6] Jong-June Jeon and Yongdai Kim. Revisiting the bradley-terry model and its application to information retrieval. *Journal of the Korean Data and Information Science Society*, 24(5):1089–1099, 2013.

[7] Kenneth Lange, Eric C Chi, and Hua Zhou. A brief survey of modern optimization for statisticians. *International Statistical Review*, 82(1):46–70, 2014.

[8] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2005.

[9] Guido Masarotto, Cristiano Varin, et al. The ranking lasso and its application to sport tournaments. *The Annals of Applied Statistics*, 6(4):1949–1970, 2012.

[10] Louis L Thurstone. A law of comparative judgment. *Psychological review*, 34(4):273, 1927.

[11] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[12] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

Abstract

Soohwan Jo
The Department of Statistics
The Graduate School
Seoul National University

Ranking data is a relevant topic in many instances. A popular example is ranking contestants in sport tournaments. The Luce model is one of the most popular models for estimating ranks. It is known that teams with similar abilities can be classified into the same group through fusing parameters in the Luce model. In this study, we investigate the abilities of groups of teams by using a group penalty function. We apply the proposed method to ranking of the teams of the National Football League 2015. We confirm that the proposed method makes it easier to interpretate ranks.

Keyword :*Luce model, Ranking model, Pairwise penalty, Group penalty*
Student Number : 2015-20313