creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

이학석사 학위논문

근위 거리 알고리즘을 이용한 등위회귀함수 추정

Estimation for Isotonic Regression using the Proximal Distance Algorithm

2019년 2월

서울대학교 대학원

통계학과

정지한

근위 거리 알고리즘을 이용한 등위회귀함수 추정

Estimation of Isotonic Regression using the Proximal Distance Algorithm

지도교수 원 중 호

이 논문을 이학석사 학위논문으로 제출함

2018년 12월

서울대학교 대학원

통계학과

정 지 한

정지한의 이학석사 학위논문을 인준함

2018년 12월

위 원 장	Myunghee Cho Paik	(인)
부위원장	원 중 호	(인)
위 원	임 채 영	(인)

국문초록

본 논문에서는 부분적인 순서 제약 조건하에서 등위회귀함수를 추정하는 문제를 푸는 알고리즘을 다룬다. 손실함수가 오차제곱함수일 경우, 이 문제는 이차계획법 문제가 되고 재귀적 분할법을 이용한 효율적인 알고리즘이 알려져 있다. 손실함수가 미분가능한 임의의 볼록함수인 경우에 대한 일반화도 제안되어 있으나, 실제 적용해본 결과, 후버 손실함수에서 오류가 발생하였고, 절대오차 손실함수나 힌지 손실함수와 같이 부분적으로 미분이 불가한 함수에 대해서는 적용하지 못하는 한계가 있었다. 이 논문에서는 먼저 일반화된 재귀적 분할 알고리즘이 후버 손실함수에서 발생하는 문제와 원인을 제시하고, 실제로 적용가능한 함수의 범위를 제시하였다. 다음으로, 상계최소화법의 일종인 근위 거리 알고리즘을 통하여, 기존의 풀리지 않던 손실함수들에 대해서도 적용 가능한 알고리즘을 제안하였다. 더불어, 근위 거리 알고리즘 구현 과정에서 대리함수의 최적해를 순서 제약이 있는 집합으로 사영시킬 때 사용될 2개 알고리즘의 이론상 속도를 비교하였고, 시뮬레이션을 통해 실제 성능을 제시하였다. 마지막으로, 절대오차 손실함수와 후버 손실함수하에서 근위 거리 알고리즘을 사용해 등위회귀함수를 추정하는 예를 제시하였다.

주요어: 등위회귀함수, GIRP 알고리즘, 파라메트릭 탐색 알고리즘, 상계최소화법, 근위 거리 알고리즘

학번: 2017-26998

Contents

1 서론 **7**

1.1 등위회귀함수 추정 7

1.2 이전 업적들 8

1.3 기여 10

2 기존 알고리즘 소개 **12**

2.1 파라메트릭 탐색 알고리즘 12

2.2 GIRP 알고리즘 17

3 GIRP 알고리즘의 문제점 **21**

4 근위 거리 알고리즘 **27**

5 실험결과 **30**

5.1 기존 알고리즘 비교 30

5.2 근위거리 알고리즘 구현 31

5.3 실제 자료 적용 사례 33

5.3.1 이상점 보정하기 33

6 결론 **36**

부록 **37**

List of Tables

3.1 반례 자료 21
3.2 반례 자료의 순서 제약 조건 22

5.1 두 알고리즘의 구현 속도 비교(단위 : 초) 31
5.2 손실함수 정리 32
5.3 포아송 분포 시뮬레이션 자료에 적용한 사례 : 평균(표준편차) 33
5.4 정규분포 시뮬레이션 자료에 적용한 사례 : 평균(표준편차) 33
5.5 자동차 자료 이상점 추가시 검증자료 MSE 34

List of Figures

1.1 오차제곱 손실함수 하 등위회귀함수 추정 예시 8

3.1 GIRP 알고리즘의 문제 상황 23

5.1 이상점 있을 때의 학습자료 예측결과 35

Chapter 1

서론

1.1 등위회귀함수 추정

등위회귀함수는 순서 제약에 대해 단조증가하는 성질을 갖는 회귀함수이다. 자료 (x_i, y_i), $x_i \in \mathbb{R}^p$, $y_i \in \mathbb{R}, (i = 1, ..., n)$으로부터 등위회귀함수를 추정하는 것은 다음과 같은 제약조건을 만족하는 $\hat{y}_1, ..., \hat{y}_n$을 추정하는 것을 의미한다.

$$\begin{aligned} &\min\{l(\hat{y}, y) \text{ subject to } \hat{y}_i \leq \hat{y}_j \, , \forall (i,j) \in C\} \\ &\text{where } \; C = \{(i,j) : x_i \preceq x_j\} \end{aligned} \tag{1.1}$$

여기서 $\hat{y} = (\hat{y}_1, .., \hat{y}_n)$, $y = (y_1, ..., y_n)$이며, $l(\hat{y}, y)$는 적합을 위한 손실함수를 의미하고, 오차제곱함수를 사용하는 경우 $l(\hat{y}, y) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$이 된다. 부분 순서 $x_i \preceq x_j$는 $x_{ik} \leq x_{jk}$, $\forall k = 1, ..p$를 의미한다. 이때, 순서 제약 조건이 담긴 집합 C에는 순서 제약이 모든 (x_i, x_j)쌍에 대해 정의되는 경우와, 자료 일부에 대해 정의되는 경우로 크게 2가지 유형이 있는데, 이후 소개되는 방법은 2가지 유형 모두에 적용되는 방법이다. 순서 제약이 없는 경우에는 오차제곱 손실함수를 최소화하는 추정량은 $\hat{y} = y$로, 회귀함수가 과적합된다. 해가 순서 제약을 만족해야 하는 경우, 최종 해는 n차원 벡터공간에서 순서제약을 만족하는 공간으로의 사영이 된다.

그림 1.1은, $p = 1$이고, 완전순서제약 하에서 손실함수로 오차제곱함수를 적용하였을 때, 최적의 추정량(실선)을 실제 자료(점선)와 함께 그린 그림이다. 실선은 등위회귀함수를 의미하는데, 계단형으로 단조증가하는 형태로 나타난다.

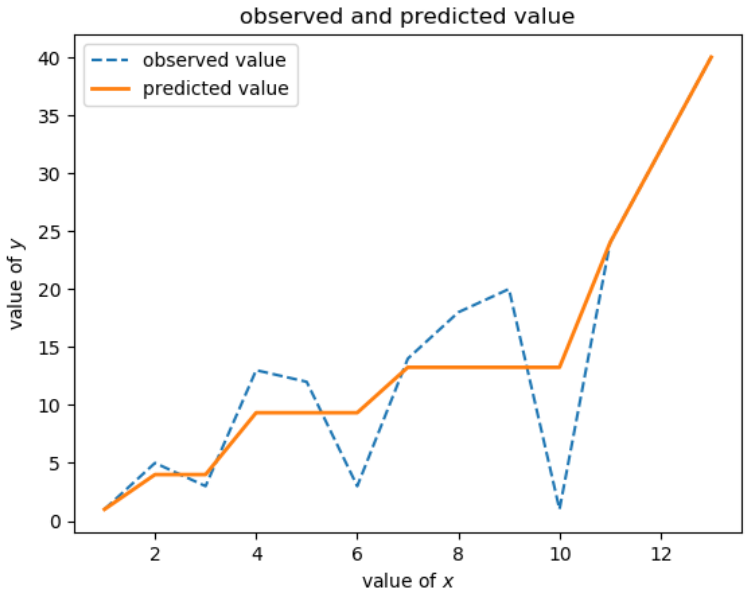


Figure 1.1: 오차제곱 손실함수 하 등위회귀함수 추정 예시

등위회귀분석은 여러 분야에서 사용되는데, 그 중에 다차원척도법과 이상점의 영향을 배제하는 회귀선 추정에서 사용되는 사례가 있다. 다차원척도법에서는 원래 고차원에 있던 자료들을 저차원 형태로 만드는 과정에서 저차원에서 대응된 값들이 원래 고차원에서의 순서 제약조건을 만족시켜야 할 경우가 있는데, 고차원에서 저차원으로의 사상에 해당하는 등위회귀함수를 추정하여 순서 제약을 유지하는 저차원 값들을 계산해 낼 수 있다. 또한 이상점이 존재하는 경우, 보통의 회귀선은 이상점에 영향을 받아 왜곡된 회귀선이 추정될 수 있는데, 이상점의 값을 제약하는 순서제약을 추가해줌으로써 이상점에 영향을 받지 않는 등위회귀함수를 추정할 수 있다.

1.2 이전 업적들

지금까지 등위회귀함수를 추정하는 여러 알고리즘이 제안되었는데, 기본적으로 미분가능한 함수나 볼록함수를 손실함수로 갖는 경우에 적용할 수 있는 방법들이다. 본 논문에서는 이 중에서 파라메트릭 탐색 알고리즘과 일반화 등위 재귀적 분할(generalized isotonic recursive partitioning algorithm, 이하 GIRP) 알고리즘만을 다루었다.

먼저, Hochbaum and Queyranne (2003)는 블록 손실함수하에서 함수값으로 정수만을

갖고, 각각의 함수값마다 하한값과 상한값의 제약이 있는 상황에서 등위회귀함수를 추정하는 파라메트릭 탐색 알고리즘을 제안하였다. 이 문제는 아래 수식처럼 표현되는데, 볼록 손실함수 클로저(convex cost closure)문제라고 불린다. 이후 논의에서, $l_i(\hat{y}_i) = l(\hat{y}_i, y_i)$를 의미한다.

$$\min\{l(\hat{y}, y) : \hat{y}_i \leq \hat{y}_j \ , \forall (i,j) \in C \text{ and } l_k \leq \hat{y}_k \leq u_k \text{ for } k = 1, ..., n\} \tag{1.2}$$
$$\text{subject to } \ C = \{(i,j) : x_i \preceq x_j\}, \quad l(\hat{y}, y) = \sum_{i=1}^{n} l_i(\hat{y}_i), \quad l_i\text{는 볼록함수}$$

여기서 (l_k, u_k)는 k번째 자료의 하한과 상한의 조합이다. 이는 알고리즘을 실행하기 전에 정해준다. 이 방법은 dynamic-tree push-relabel 최소절단 알고리즘에 기반을 두고 있다. 이 알고리즘은 등위회귀함수의 함수값을 추정하는데 사용되는 중단점들을 분할 탐색 및 재귀방식을 통하여 찾아낸다. 특정 추정값을 가정한 상태에서 결과물로 소스 집합과 여집합인 타겟 집합이 만들어지는데, 추정값이 커질 때마다 소스 집합은 같거나 커지고, 타겟 집합은 같거나 작아진다. 이때, 소스 집합이 커지는 경우에 가정된 추정값이 중단점에 해당하고, 이를 저장하여, 모든 작업이 종료되었을 때, 추정값을 계산하는데 사용한다. 이 방법은 볼록함수에 대해 사용할 수 있지만, 기본적으로 정수해만 구할 수 있는 한계가 있다.

비교적 최근의 방식은 Luss and Rosset (2014)가 제안하였는데, Karush-Kuhn-Tucker(이하 KKT) 조건 (Boyd and Vandenberghe, 2004)에 기반하여 도출된 GIRP 알고리즘이다. 이 알고리즘은 앞선 파라메트릭 탐색 알고리즘보다 더 제한된 조건인 미분 가능한 볼록함수에 대해서 사용가능하다고 알려져 있다.

$$\min\{l(\hat{y}, y) \text{ subject to } \hat{y}_i \leq \hat{y}_j \ , \forall (i,j) \in C\} \tag{1.3}$$
$$\text{where } \ C = \{(i,j) : x_i \preceq x_j\}, \quad l(\hat{y}, y) = \sum_{i=1}^{n} l_i(\hat{y}_i), \quad l_i\text{는 미분가능한 볼록함수}$$

이 알고리즘의 주요 아이디어는 기존의 집합을 각각의 집합 내 원소들의 미분계수값들의 부분합 차가 가장 작도록 하는 2개의 서로 배반된 집합으로 분할하는 방식이다. 이 문제는 선형 계획법을 이용하여 풀 수 있다. 이러한 분할을 더이상 분할되지 않을 때까지 진행하는데, 분할 과정에서 도출되는 해들이 순서제약 조건을 만족한다는 점이 큰 특징이다. 또한, 앞선 파라메트릭 탐색 알고리즘과 다르게, 실수해를 도출할 수 있다는 장점이 있다.

두 알고리즘은 적용 가능한 함수의 범위와 최적해의 형태 측면뿐만 아니라 속도 측면에서도 차이가 있다. 이론상으로는, 파라메트릭 탐색 알고리즘이 시간 복잡도 측면에서 GIRP

알고리즘보다 효율적인 것으로 나온다. 후술하겠지만, 실제로 적용해본 결과 파라메트릭 탐색 방식은 실용적인 문제가 있었고, 속도와 안정성 측면에서 GIRP 알고리즘에 뒤쳐졌다.

1.3 기여

위에서 소개된 2개의 알고리즘은 각자 한계를 가지고 있다. 파라메트릭 탐색 알고리즘은 볼록함수에 적용가능하나, 기본적으로 해가 정수여야 한다는 제약이 있고, GIRP 알고리즘은 미분가능한 볼록함수에만 적용 가능하다. 또한, 원 논문의 주장과는 다르게 GIRP 알고리즘은 후버 손실함수(Huber et al., 1964)에 대해 순서제약을 만족하는 해를 도출하지 못할 수가 있다. 그렇기에, 실수해를 가지면서 미분 불가능한 절대오차 손실함수나, 힌지(hinge) 손실함수(Wahba, 1999), 또는 후버 손실함수를 적용하는 경우에는 GIRP 알고리즘을 바로 적용하기 힘들고, 파라메트릭 탐색 알고리즘도 정확한 해를 구하는데 사용되기 힘들다. 결론적으로, 실수해를 구하면서, 일반적인 형태의 볼록한 손실함수에 대해 적용 가능한 방법을 제시하고자 하는 것이 이 논문의 목표이며, 이에 대한 방안으로 근위 거리 알고리즘을 도입하였다.

본 논문에서는 상계최소화법(majorization-minimization algorithm, 이하 MM 알고리즘)의 일종인 근위 거리 알고리즘을 도입하여, 기존의 알고리즘을 적용하기는 힘들지만 근위사상을 적용할 수 있는 손실함수 하에서 등위회귀함수를 추정하는 방법을 제시하였다. 이를 통해 기존 논문의 알고리즘으로 해결이 불가능했던, 절댓값 손실함수, 후버 손실함수, 힌지 손실함수 하에서의 등위회귀함수 추정이 가능해졌다. MM 알고리즘은 목적 함수의 최솟값이 되는 해를 찾을 때, 그 함수의 상계가 되는 대리함수를 찾은 뒤, 그 대리함수를 반복적으로 최소화하는 방식으로 해를 찾는 알고리즘이다. 근위 거리 알고리즘은 해와 주어진 제약 집합 사이의 거리함수에 대한 대리함수를 이용하는 MM 알고리즘이다. 구체적으로는, 이전 작업의 최적해를 바탕으로 만들어진 새로운 대리함수의 근위사상을 구하고, 근위사상의 결과물을 제약 집합으로 사영시키는 작업을 반복한다. 등위함수 추정을 위해서는, 대리함수의 근위사상으로 도출된 값을 순서 제약의 집합으로 사영시키는 과정에서 오차제곱함수에 대한 기존의 알고리즘을 사용하였다.

이 논문은 다음과 같은 순서로 구성되어 있다. 2장에서는 앞서 소개한 두 알고리즘의 구체적인 작동 방식에 대해서 자세히 설명하겠다. 이어 3장에서는 후버 손실함수에서 GIRP

알고리즘이 작동하지 않는 반례를 제시하고, Luss and Rosset (2014)의 Theorem 1 증명의 잘못된 부분을 지적한 뒤, GIRP 알고리즘이 순볼록 함수(strictly convex function)이면서 강압적 함수(coercive function)에서만 오류없이 작동함을 보이겠다. 4장에서는 근위 거리 알고리즘을 소개하고, 이를 이용해 등위회귀함수 추정 문제를 푸는 과정을 제시하겠다. 이어서, 5장에는 시뮬레이션을 통해 근위 거리 알고리즘이 잘 작동함을 보이고, 실제 데이터에 근위 거리 알고리즘을 적용한 결과물을 제시하겠다. 마지막으로 6장에서는 앞선 내용들을 정리하고, 제언을 하도록 하겠다. 이 논문에서 사용된 코드는 줄리아(Julia) 언어(Bezanson et al., 2012)로 작성되었으며, 논문에 사용된 모든 코드를 깃허브[1]에 올려두었다. 또한, 본 논문에서 사용된 여러 손실함수의 근위사상 도출 과정을 부록에 담아두었다.

[1] https://github.com/Jihan-Jung/JH_Isotonic_regression

Chapter 2

기존 알고리즘 소개

이 장에서는 앞에서 제안된 2가지 알고리즘의 구체적인 작동 방법을 다루고, 각 알고리즘별로 시간복잡도를 제시한다.

2.1 파라메트릭 탐색 알고리즘

파라메트릭 탐색 알고리즘(Algorithm 1)은 재귀방식으로 특정한 추정값 λ가 주어졌을 때, 다음과 같은 서브작업을 시행한다. 서브작업은 각 자료들을 노드(V 집합)로 하고, 추가적으로 소스노드(s)와 타겟노드(t)를 만든 뒤, 근사된 미분계수와 순서 제약을 이용해 노드 간에 엣지(연결선)를 갖는 그래프를 만들고, 최소절단(mincut)이 되는 2개의 집합으로 나누는 작업을 말한다. 최소절단 집합이란, 두 개의 집합으로 쪼갰을 때, 집합 간의 용량 차이가 가장 작은 집합을 의미한다. 집합 간의 용량 차이란, 두 집합 중 한 집합에 들어오는 방향의 엣지들의 용량합과 다른 집합으로 나가는 방향의 엣지들의 용량합의 차이를 의미한다. 이때, 두 집합은 각각 소스노드와 타겟노드를 포함하는데, 소스노드를 포함하는 집합을 소스집합이라고 한다.

서로 다른 λ에서 진행되는 서브작업에서 바뀌는 것은 엣지의 연결 방식과 용량들인데, 이것을 결정하는 것은 특정한 추정값이 λ로 주어졌을 때의 각 자료별로 계산된 미분계수의 근사값들이다. j번째 노드에서의 미분계수의 근사값은 $l'_j(\lambda) = l_j(\lambda) - l_j(\lambda - 1)$으로 정의된다. 각 노드별로 이 근사된 미분계수가 양수인 경우에는 소스노드와 노드를 연결하고, 소스로부터 노드까지의 엣지를 만들어준 뒤, 엣지의 용량을 미분계수값으로 한다. 반대로,

음수인 경우에는 자료노드와 타겟노드를 연결하고, 엣지의 용량을 미분계수값의 절댓값으로 한다. 크기 순서는 용량이 무제한인 엣지를 크기가 큰 노드에서 크기가 작은 노드 방향으로 연결시키는 방식으로 구현한다. λ가 커질수록 최솟값이 λ보다 작은 노드들이 많아지고, 이는 근사된 미분계수값이 양수가 됨을 의미한다. 그렇기에 최소절단을 진행하기에 앞서 소스집합은 점점 커지는 경향을 보이게 된다.

최대흐름(maxflow)은 그래프에서 정의된 엣지들의 용량하에서 소스노드에서 타겟노드로 보낼 수 있는 가장 큰 흐름을 의미한다. 한편, 최대흐름을 구하는 것이 최소절단을 구하는 것과 동치임이 알려져 있다(Cormen et al., 2009). 최대흐름이 소스노드에서 타겟노드로 진행하는 과정에서 중간에 용량이 작은 부분에 의해서 그 양이 한정되기 때문에, 최대흐름을 구현하고, 남아있는 용량을 나타내는 그래프에서 연결부분이 끊긴 곳들이 용량 차이가 작은 부분을 의미한다. 그렇기에 최대흐름을 구현한 뒤, 각각의 노드와 연결된 엣지의 남아있는 용량을 기반으로 소스노드와 타겟노드 중 어디에 연결되어 있는지를 파악하면, 최소절단을 구할 수 있게 된다. 특정 λ에서 최소절단으로 구한 소스집합을 S_λ라고 정의하면, 앞서 언급한 성질에 의하여, $S_\lambda \subseteq S_{\lambda+1}$이 되며, 이를 귀납적으로 적용하면 일렬의 소스집합 간의 크기관계가 도출된다. 이때, $S_{\lambda_1} = \{s\}$인 λ_1 이후에 최초로 어떤 λ_2에서 집합 간의 종속관계가 생긴다면, 즉 $S_{\lambda_1} \subset S_{\lambda_2}$가 최초로 성립한다면, 이 값을 중단점이라고 한다. 이런 방식으로 다시 λ_2보다 큰 값에서 최초로 어떤 값 λ_3에서 다시 종속관계가 생긴다면 이 값 역시 중단점이 된다. 이렇게 $S_{\lambda_l} = \{s\} \cup V$가 될 때까지 나온 λ_1,...,λ_l과 그에 상응하는 소스집합들은 다음과 같은 성질을 만족한다.

$$\{s\} = S_{\lambda_1} \subset S_{\lambda_2} \subset \cdots \subset S_{\lambda_l} = \{s\} \cup V \tag{2.1}$$
$$\text{where } \lambda_1 < \cdots < \lambda_l$$

파라메트릭 탐색 알고리즘은 결국 이런 중단점들을 효과적으로 찾는 방법을 의미하고, 중단점들과 소스집합이 커졌을 때 새로 추가된 노드들을 저장한다.

Algorithm 1을 보면, 파라메트릭 탐색 알고리즘은 특정 λ에 대해 최소절단을 구하는 작업과 이때, λ가 중단점이 되는지 여부를 판단하는 부분으로 나뉜다. 이때, λ 값을 하나씩 늘려나가는 방식이 아닌, λ가 속하는 구간을 입력값으로 받아, 그 구간의 중앙값에 해당하는 값에 대해 최소절단을 수행함으로써, 최대한 λ의 탐색구간을 줄이고자 한다. 그리고 중앙값을 기준으로 나뉘어진 두개의 구간에 대해 각각 파라메트릭 탐색 알고리즘을 적용하

여, 재귀적으로 작동하도록 한다. 결국 이 알고리즘은 우선적으로 최소절단을 효과적으로 할 수 있는 방법이 필요하고, 이를 위해 최대흐름을 구현하는 dynamic push-relabel 알고리즘(Algorithm 2)을 활용하였다. push-relabel 알고리즘은 우선 각각의 노드에서 연결된 엣지의 용량만큼 최대한 다른 노드로 보내고, 노드에 잉여의 용량들을 한 단계 낮은 label의 노드로 계속해서 보내준다(push). 만약 주변의 연결된 노드가 label이 서로 같다면 높이를 하나 높인 뒤(relabel) 다시 잉여의 용량들을 주변의 높이가 1 낮은 노드로 보내는 작업을 수행한다. 그리고, 이런 노드간의 라벨차이를 반영한 트리구조를 만들어 저장하고, 이를 이용해 위의 잉여 용량을 보내고, 해당 노드의 높이를 1 높이는 작업을 쉽고 빠르게 수행한다. 그 다음 중단점을 저장할지에 관한 판단을 하는 부분이 있는데, Gallo et al. (1989)에서 이 부분에 대한 설명이 잘 되어있지 않고 모호하게 표현되어 있어 중단점의 정의에 따라 현재 λ보다 1 작은 값에 대한 소스집합을 구하고, 현재의 소스집합과 비교해서 중단점 여부를 판단하였다.

재귀적인 방식으로 모든 λ를 탐색하지 않고, 효과적으로 중단점을 찾는 것이 이 알고리즘이 달성하고자 하는 목표이다. 또한 λ가 작은 서브작업에서 계산된 결과 중 노드별 높이 정보와 엣지별 보내지는 용량에 관한 정보를 이보다 큰 λ값을 갖는 상황에서 활용될 수 있기에 점점 서브작업에 소요되는 시간이 적어진다. 이렇게 저장된 중단점들을 바탕으로 중단점 λ_k에서 새로이 추가된 원소들을 담은 $S(k) = S_{\lambda_k} - S_{\lambda_{k-1}}$의 원소들에 대해 추정값을 $\lambda_{k-1} + 1$로 한다.

이론상으로, 이 알고리즘은 이전의 분석결과를 저장해서 추후에 지속적으로 활용한다는 점에서 효율적이고, 최소절단 집합을 구하는 dynamic push-relabel 알고리즘이 작동 속도가 빠르다고 알려져있다. 실제 이 알고리즘의 이론상 시간 복잡도는 $O(mn \log n^2/m)$으로 알려져있다.

Algorithm 1 파라메트릭 탐색 알고리즘

Require: 자료가 $(x_1, y_1), ..., (x_n, y_n)$, $Y = (y_1, y_2, ...)$ 주어지고, 부분적으로 순서 제약 공간(C)이 주어진 경우

Require: 중단점 후보 구간 (λ_1,λ_2)과 λ_1,λ_2 각각의 소스집합(타겟집합) $S_{\lambda_1}(\bar{S}_{\lambda_2})$,$S_{\lambda_2}(\bar{S}_{\lambda_2})$

Require: 소스노드는 s, 타겟노드는 t

$s = \{s\} \cup S_{\lambda_1}$, $t = \{t\} \cup \bar{S}_{\lambda_2}$로 합치고, 남은 노드들을 V의 원소라고 한다.

if $V = \{s, t\}$ or $\lambda_2 - \lambda_1 \leq 1$ **then**

    중단점 없기 때문에 종료

end if

$\lambda^* = \lfloor \frac{\lambda_1 + \lambda_2}{2} \rfloor$

λ^*에서 그래프를 만들고, 최소절단을 수행하여 S_{λ^*}를 도출

if λ^*가 중단점 **then**

    λ^*와 S_{λ^*}를 저장

end if

재귀적으로 (λ_1,λ^*) 와 S_{λ_1},S_{λ^*} 에 대해 알고리즘 적용

재귀적으로 (λ^*,λ_2) 와 S_{λ^*},S_{λ_2} 에 대해 알고리즘 적용

Algorithm 2 Dynamic Tree Push Relabel 알고리즘

Require: 노드 간의 연결 상태를 저장한 treePath, 엣지의 남은 용량을 반영한 residual_graph

Require: 용량이 초과한 노드들을 담은 Q, 현재의 pushrelabel이 적용될 노드 v

1: v에서 플로우를 보낼 수 있는 주변 노드들을 찾는다.
2: **for** i번째 주변노드 **do**
3: 노드 v의 용량이 초과되지 않았다면 반복문을 종료한다
4: **if** 노드 v의 높이가 i번째 주변노드 높이보다 1 클 때 **then**
5: 보낼 수 있는 만큼 노드 v의 초과된 용량을 i번째 노드로 보냄, treePath에 변화된 정보 저장
6: **else**
7: **if** 마지막 주변 노드가 아닐 때 **then**
8: 2행으로 다시 가서 (i+1) 주변노드에 대해 작업 수행
9: **else**
10: 현재 노드를 relabel(높이 1 증가시키기)하고, treePath에 변화된 정보 저장
11: **end if**
12: **end if**
13: **end for**

2.2 GIRP 알고리즘

GIRP 알고리즘은 전체 자료가 담긴 집합에서, 순서 제약을 만족하면서 전체 손실함수 값을 최소로 하기 위해 추정값이 더 커져야할 원소들을 담은 집합과 작아져야할 원소들을 담은 집합으로 분할하는 작업을 수행한다. 그리고, 분할된 모든 집합에 대해 같은 방식의 분할 작업을 반복하여 어떤 집합도 더 이상 분할되지 않을 때까지 알고리즘이 수행된다.

자료 (x_i, y_i), $x_i \in \mathbb{R}^p$, $y_i \in \mathbb{R}$ $(i = 1, ..., n)$가 주어졌을 때, 전체 손실함수 l이 $l(\hat{y}, y) = \sum_{i=1}^{n} l_i(\hat{y}_i)$처럼 자료 각각에 대한 손실함수인 l_i들의 합으로 표현되고, l_i가 모든 i에서 미분가능한 볼록함수일 때 GIRP 알고리즘을 통해 (2.2)를 만족하는 최적해를 구할 수 있다고 알려져있다. $l_i(\hat{y}_i) = l(\hat{y}_i, y_i)$, $\hat{y} = (\hat{y}_1, .., \hat{y}_n)$, $y = (y_1, ..., y_n)$을 의미한다.

$$\min\{\sum_{i=1}^{n} l_i(\hat{y}_i) \text{ subject to } \hat{y}_i \le \hat{y}_j \ , \forall (i,j) \in C\} \tag{2.2}$$
$$\text{where } \ C = \{(i,j) : x_i \preceq x_j\}$$

GIRP 알고리즘의 분할 방식은 순서 제약에 대한 벌점을 추가한 손실함수를 최소로 하는 최적해들이 만족해야하는 KKT 조건들을 만족하게 만든다. 목적함수가 미분가능한 볼록함수이고, 부등식 제약함수가 선형이면, KKT 조건은 (2.2)식을 만족하는 최적해가 될 필요충분조건이 되므로, GIRP 알고리즘은 (2.2)식을 만족하는 최적해를 도출할 수 있다 (Lange, 2016). 우선 (2.2)를 라그랑지안 식으로 나타내면 다음과 같다.

$$\min\{\sum_{i=1}^{n} l_i(\hat{y}_i) + \sum_{(i,j)\in C} \lambda_{ij}(\hat{y}_i - \hat{y}_j)\} \tag{2.3}$$

이때, KKT 조건에 의하여, 최적해가 만족해야할 조건은 다음의 4개이다. 여기서 V는 전체 자료의 인덱스를 담은 집합이고, $\hat{y}_i^*$는 (2.3)을 만족하는 최적해이고, λ_{ij}^*는 (2.3)을 만족하는 최적의 듀얼변수(dual variable)값이다.

$$(a)\frac{\partial l_i}{\partial \hat{y}_i}(\hat{y}_i^*) = \sum_{j:(j,i)\in C} \lambda_{ji}^* - \sum_{j:(i,j)\in C} \lambda_{ij}^*, \qquad \forall i \in V \tag{2.4}$$
$$(b)\hat{y}_i^* \le \hat{y}_j^*, \qquad \forall (i,j) \in C$$
$$(c)\lambda_{ij}^* \ge 0, \qquad \forall (i,j) \in C$$
$$(d)\lambda_{ij}^*(\hat{y}_i^* - \hat{y}_j^*) = 0, \qquad \forall (i,j) \in C$$

먼저 조건 (a)를 보면 $\sum_{i\in V}\frac{\partial l_i}{\partial \hat{y}_i}(\hat{y}_i^*)=0$이 성립함을 알 수 있다. 또한 조건 (d)를 보면 λ_{ij}^*가 양수일 경우, 등식이 성립하기 위해서는 $\hat{y_i}^*=\hat{y_j}^*$를 만족해야 함을 알 수 있다. 또한 조건 (b)를 보면, 최적해가 순서 제약 조건을 만족해야한다.

이러한 조건들을 GIRP 알고리즘은 다음과 같은 방식으로 달성한다. 분할이 될 가능성이 있는 자료점들의 집합 V에 대해 $\sum_{i\in V}\frac{\partial l_i}{\partial \hat{y}_i}(z)=0$을 만족하는 해 z를 구하고, 이를 w_V라고 정의하자. 이때 V의 임의의 원소 i에 대해서 $\frac{\partial l_i}{\partial \hat{y}_i}(w_V)$는, 0보다 작거나 같은 경우나 0보다 큰 경우가 있다. 이때 미분이 0보다 작은 경우는 아직 l_i가 감소하고 있으며, w_V보다 더 큰 값에서 손실함수의 값이 더 작아짐을 의미하고, 0보다 큰 경우는 반대로 w_V보다 더 작은 값에서 손실함수의 값이 더 작아짐을 의미한다. 이를 통해, V의 원소 중 전자의 경우에 해당하는 원소들을 집합 V^+에 모으고, 후자의 경우를 집합 V^-에 모은다면, 아래와 같은 식이 도출된다. 집합 X에 대해 $w_X=\underset{z}{\operatorname{argmin}}\sum_{i\in X}l_i(z,y_i)$을 의미한다.

$$\sum_{i\in V^+}\frac{\partial l_i}{\partial \hat{y}_i}(w_V)-\sum_{i\in V^-}\frac{\partial l_i}{\partial \hat{y}_i}(w_V)<0 \tag{2.5}$$

결국, 위 식의 왼쪽항이 최소가 되도록, V를 분할하면 전체 손실함수를 최소로 만들 수 있다. 이때, 순서 제약 조건을 추가하면 아래의 정수계획법 문제가 된다.

$$\min\ \{z^tx : x_i\le x_j\ \forall(i,j)\in C,\ x_i\in\{-1,1\},\ \forall i\in V\}\ where\ \ z_i=\frac{\partial l_i}{\partial \hat{y}_i}(w_V) \tag{2.6}$$

식 (2.6)의 정수제약 조건 $x_i\in\{-1,1\}$을 $-1\le x_i\le 1$ 조건으로 완화하면, 아래의 선형계획법 문제를 얻는다

$$\min\ \{z^tx : x_i\le x_j\ \forall(i,j)\in C,\ -1\le x_i\le 1,\ \forall i\in V\}\ where\ \ z_i=\frac{\partial l_i}{\partial \hat{y}_i}(w_V) \tag{2.7}$$

이때, 이 문제의 해가 정수계획법 문제 (2.6)의 해와 일치함이 잘 알려져있다 (Murty, 1983). 선형계획법 (2.7)로부터 도출된 2개의 집합은 따로 저장되었다가 이후에 분할될 집합의 후보가 된다. 만약 어떤 집합이 분할이 안되는 경우에, 이 집합을 따로 저장해 두고 이후에는 분할하지 않는다. 최종적으로 모든 집합이 분할이 될 수 없을 때까지 작업을 반복한다. 정리하면, 최종적으로 분할된 집합들은 집합의 분할 방식에 의해 조건 (b)의 순서 제약을 만족하고, 최종적으로 분할된 각각의 집합들에서의 추정값은 미분계수가 0이 되는 해가 되기 때문에, $\sum_{i\in V}\frac{\partial l_i}{\partial \hat{y}_i}(\hat{y}_i^*)=0$이 만족되고, 조건 (c),(d)가 성립하게 된다. 이를 통해 KKT 조건을 충족하는 최적해를 찾을 수 있게 된다. 위의 서술된 내용을 바탕으로 Algorithm 3

에 GIRP 알고리즘의 의사 코드를 표시하였다.

GIRP 알고리즘의 장점은 한 번 분할된 집합은 항상 최적의 분할로 가는 과정이기에 이후 과정에서 분할 방식이 바뀔 가능성이 전혀 없고, 매 분할이 이루어진 시점마다 그 시점까지 분할된 집합들은 항상 순서 제약조건을 충족시키고 있다는 점이다. Luss and Rosset (2014)에서는 이러한 성질을 활용하여, 모든 집합이 분할이 안 되는 마지막 시점까지 집합을 분할함으로써 생기게 되는 과적합을 방지하고자, 매 분할마다 검증자료에 대해 평균제곱오차와 같은 추정량의 오차를 계산하여, 그 값이 최소가 되도록 분할하는 방법도 제안하고 있다. 이 알고리즘의 경우, 파라메트릭 탐색 알고리즘과 달리 이전 분석결과를 활용하여 속도를 개선할 방법이 없고, 매 분할마다 선형 계획법 문제를 풀어야하며, 집합 내 원소들을 모두 살펴봐야 하는 작업이 수반되어 시간 복잡도가 $O(n^3)$으로 알려져있다. 하지만 실제 구현 결과, 시뮬레이션 자료에서는 파라메트릭 탐색 알고리즘보다 더 빠르게 작동하였다.

Algorithm 3 GIRP 알고리즘

Require: 자료가 $(x_1, y_1), ..., (x_n, y_n)$으로 주어지고, 순서 제약 집합$(C)$이 주어진 경우

Require: $A = \{(1, 2, ..., n)\}$, $B = \varnothing$, $S = \{(0, \{1, 2, ..., n\}, \{\})\}$. 어떤 집합 X에 대해 $w_X = \underset{\hat{y}}{\operatorname{argmin}} \sum_{i \in X} l_i(\hat{y})$로 정의하면, 집합을 원소로 가지는 집합 $Q = \{Q_1, ..., Q_m\}$이라고 할 때, $Y_Q = (w_{Q_1}, ..., w_{Q_m})$으로 정의한다.

1: $k = 0$
2: **while** $A \neq \varnothing$ **do**
3: S의 원소쌍들 중에서 첫번째 값이 가장 큰 쌍을 (val, w^-, w^+)로 표현하고, 이를 선택한다.
4: $A = (A \setminus (w^- \cup w^+)) \cup \{w^-, w^+\}$, $S = S \setminus (val, w^-, w^+)$로 갱신한다.
5: k번째 스텝에서 $M[k] = (A, Y_A)$
6: **for** $v \in \{w^-, w^+\}$ **do**
7: **if** $v \neq \varnothing$ **then**
8: 6번 행으로 다시 간다.
9: **end if**
10: v에 대해 (2.5)식을 적용하여, 집합 v에 대하여 최적의 x를 구한다.
11: **if** x의 부호가 다 같다 **then**
12: $A = A \setminus w$, $B = B \cup w$
13: **else**
14: $v^+ = \{i : x_i = 1\}$, $v^- = \{i : x_i = -1\}$, $S = S \cup (z^t x, v^-, v^+)$. $z^t x$값이 클수록 더 쪼개질 가능성이 큰 집합에 해당한다.
15: **end if**
16: **end for**
17: $k = k + 1$
18: **end while**

Chapter 3

GIRP 알고리즘의 문제점

1장에서 언급한 대로, 후버 손실함수 하에서 GIRP 알고리즘을 통해 구한 해가 순서 제약 조건을 만족하지 않을 수 있다. 후버 손실함수는 미분가능한 볼록함수이고, 다음과 같이 정의된다. 여기서

$$l_i(\hat{y}_i) = \begin{cases} \frac{(\hat{y}_i - y_i)^2}{2} & |\hat{y}_i - y_i| < \delta \\ \delta(|\hat{y}_i - y_i| - \frac{\delta}{2}) & |\hat{y}_i - y_i| \geq \delta \end{cases}$$

표 3.1에 제시된 63개의 자료값과 표 3.2에 제시된 97개의 순서 제약 조건을 생각하자. 표 3.2의 순서쌍 (i, j)는 $\hat{y}_i \leq \hat{y}_j$인 제약조건을 의미한다.

Table 3.1: 반례 자료

인덱스	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
자료값	5	6	5	5	6	5	7	6	7	8	4	5	5	7	7	2	5	5	2	8
인덱스	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
자료값	2	1	5	4	7	4	6	3	6	6	2	8	6	4	5	9	3	4	6	5
인덱스	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
자료값	3	6	2	4	2	5	5	5	5	5	6	7	1	3	5	6	4	5	3	3
인덱스	61	62	63																	
자료값	5	4	3																	

Table 3.2: 반례 자료의 순서 제약 조건

순서 제약조건
(3,8) (7,9) (2,11) (3,11) (4,11) (12,13) (11,14)
(2,15) (4,15) (7,15) (4,16) (5,16) (8,16) (14,17)
(13,18) (2,19) (9,19) (13,20) (17,21) (8,22) (21,22)
(6,23) (7,23) (10,23) (9,24) (15,24) (21,24) (8,26)
(9,26) (15,26) (21,26) (4,28) (6,28) (19,28) (22,30)
(24,30) (26,30) (6,31) (10,31) (18,32) (13,33) (25,33)
(6,34) (15,34) (19,34) (6,35) (19,35) (24,35) (5,37)
(30,37) (1,38) (18,38) (20,38) (18,39) (20,39) (25,40)
(27,40) (32,41) (33,41) (36,43) (29,44) (20,45) (27,45)
(41,45) (27,46) (20,48) (41,48) (29,49) (36,49) (42,51)
(46,51) (40,53) (45,53) (42,54) (47,56) (40,57) (51,57)
(40,58) (41,58) (6,59) (10,59) (55,59) (56,59) (6,60)
(10,60) (55,60) (56,60) (6,61) (10,61) (55,61) (56,61)
(39,62) (46,62) (48,62) (53,62) (58,62) (52,63)

이 자료에 GIRP 알고리즘을 적용하여 분할된 집합들을 노드들로 설정하여, 그림 3.1에 나무구조로 표현했다. 이 때, 나무의 각 노드에 담긴 값들은 원자료의 인덱스를 의미하며, 괄호 안에 있는 값은 분할된 집합에서 최적해를 의미한다. 후버 손실함수에서의 δ값은 0.9로 하였다. 그림 3.1에서 점선 테두리를 가지는 노드가 문제가 되는 노드이다.

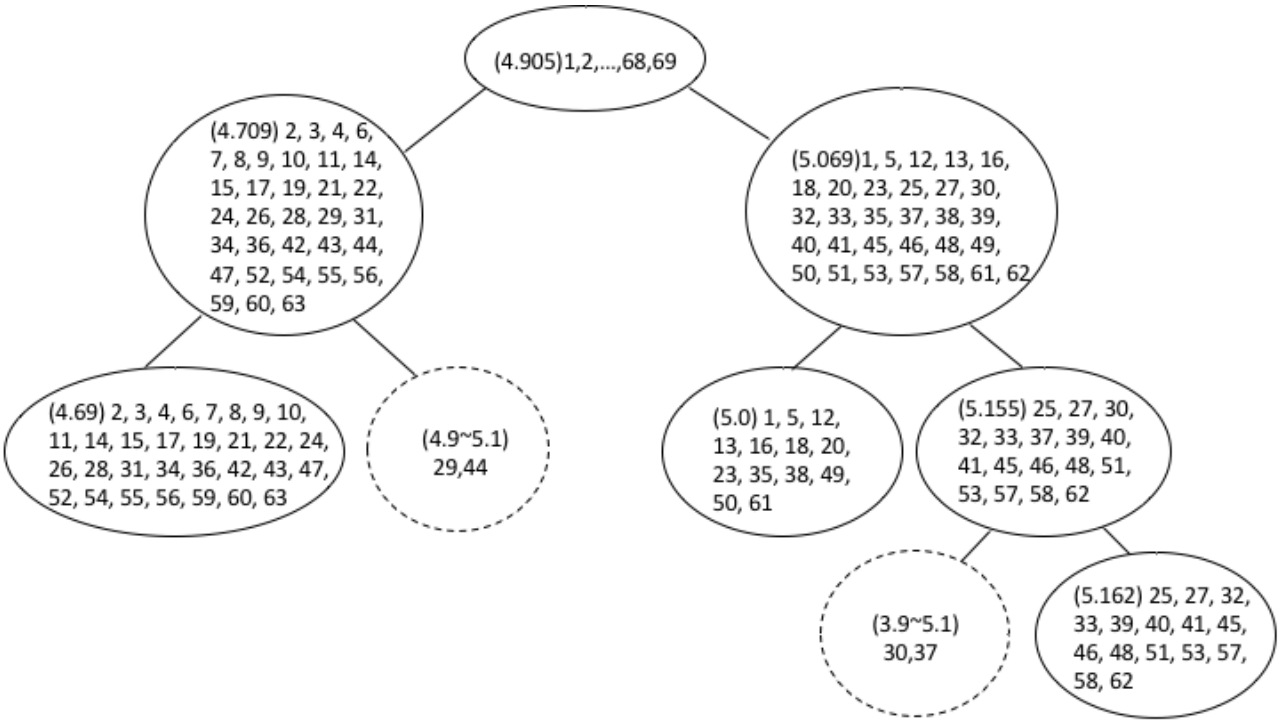


Figure 3.1: GIRP 알고리즘의 문제 상황

먼저 29번째 자료값인 6과 44번째 자료인 4를 가지는 노드는 4.9에서 5.1 사이의 모든 해가 최적해가 될 수 있다. 이 때, 해를 5.1이라고 하면 순서제약 조건을 위배하게 된다. 30번째 자료값인 6과 37번째 자료값인 3을 가지는 노드는 3.9에서 5.1 사이의 모든 해가 최적해가 될 수 있는데, 해가 3.9라고 하면 역시 순서제약 조건을 위배하게 된다. 즉, 이들 노드가 속한 서브트리(subtree)의 조상노드를 거슬로 올라가지 않으면 순서 제약을 만족하는지 확신할 수 없으며, 이는 GIRP의 재귀적 분할 방법이 손실함수가 볼록함수일 때 최적해를 준다는 주장에 위배된다.

이는 Luss and Rosset (2014)의 Theorem 1 증명에 오류가 있기 때문이다. 해당 정리의 주장과 그 증명을 먼저 아래에 제시한다. 블록은 적합치가 같은 자료점들의 집합이다. 최적블록은 문제 (2.2)의 최적해($\hat{y}^*$)로 구성한 블록을 의미한다. 즉, 최적해 $\hat{y}^* = (\hat{y}_1^*, ..., \hat{y}_n^*)$ 중에 같은 값을 갖는 원소들이 하나의 블록으로 묶인다.

Theorem 1. *V가 최적 블록들의 집합 일때, 선형계획법 문제 (2.7)식을 풀어서 얻어지는 절단(이하 알고리즘 절단)으로는 V에 속하는 임의의 최적 블록을 분할할 수 없다.*

Proof. $\tilde{V}$를 V의 원소들의 합집합이라고 정의할 때, 집합 L,U,M을 아래와 같이 정의한다.
(1) $L : \tilde{V}$의 원소 중 알고리즘 절단 아래에 있는 원소들의 집합
(2) $U : \tilde{V}$의 원소 중 알고리즘 절단 위에 있는 원소들의 집합
(3) $M : V$의 원소 중에서 알고리즘 절단에 의해 쪼개지는 블록들의 집합
여기서 알고리즘 절단 위에 있는 원소는 문제 (2.7)을 푼 결과 $x_i = 1$이 나오는 원소를 의미하고, 절단 아래에 있는 원소는 $x_i = -1$이 나오는 원소를 의미한다. 임의의 블록 X에 대해 $w_X = \underset{z}{\operatorname{argmin}} \sum_{i \in X} l_i(z)$로 정의한다. V안에 S개의 최적 블록이 있다고 할 때, 각각의 최적 블록을 $V_i (i = 1, ..., S)$으로 표현하면, $w_V = \underset{z}{\operatorname{argmin}} \sum_{j=1}^{S} \sum_{i \in V_j} l_i(z)$로 정의한다. 두 블록 A, B에 대해 $A \preceq B$는 $w_A \leq w_B$를 의미하고, $A \prec B$는 $w_A < w_B$를 의미한다. 증명은 분할되는 블록이 존재한다고 가정할 때, 모순이 발생함을 보이는 방식으로 진행된다. 가정상 분할되는 블록이 존재하므로, M의 원소 중 적어도 하나는 분할된다. 분할되는 블록들이 K개 있다고 할 때, $M_1 \preceq M_2 \preceq ... \preceq M_K$를 만족하도록, 블록 $M_i \in M$에 순서를 정한다. 이제, 블록 $M_i (i = 1, ..., K)$에서 M_i^U는 M_i에서 알고리즘 절단 위에 있는 원소들을 모은 집합이고, M_i^L은 M_i에서 알고리즘 절단 아래에 있는 원소들을 모은 집합이다. 즉 $M_i^U = M_i \cap U$이고, $M_i^L = M_i \cap L$이다. 더 나아가, U에서 가장 작은 최적 적합치를 갖는 공집합이 아닌 블록을 A_1^U로 정의하고, L에서 가장 큰 최적해를 갖는 공집합이 아닌 부분집합을 A_K^L이라고 정의하자. M_i의 정의에 의해 다음이 성립한다.

$$w_{M_1} \leq w_{M_K} \tag{3.1}$$

먼저, M_1과 M_K는 최적 블록이므로, 다음 두 식이 성립한다.

$$w_{M_1^L} > w_{M_1^U} \tag{3.2}$$

$$w_{M_K^L} > w_{M_K^U} \tag{3.3}$$

그 이유는 다음과 같다. 만약, $w_{M_1^L} < w_{M_1^U}$이 성립한다면, M_1^L과 M_1^U가 각각 최적 블록이 되어 M_1이 최적 블록이라는 가정을 위배하게 된다. $w_{M_1^L} = w_{M_1^U}$인 경우에는 분할을 해도 (2.2)에서 손실함수값이 감소하지 않기 때문에, 분할을 하지 않는다. M_K에 대해서도 같은 논리를 적용하면 $w_{M_K^L} \leq w_{M_K^U}$가 성립할 수 없다.

한편, l_i는 볼록함수여서 $\sum_{i\in A_1^U} l_i(z)$도 볼록함수이고, 볼록함수의 초평면 부등식에 의하여 아래의 식이 성립한다.

$$\sum_{i\in A_1^U} l_i(w_{A_1^U}) \geq \sum_{i\in A_1^U} l_i(w_V) + (w_{A_1^U} - w_V) \sum_{i\in A_1^U} \frac{\partial l_i}{\partial \hat{y}_i}(w_V) \tag{3.4}$$

이때, $w_{A_1^U}$의 정의에 의해, $\sum_{i\in A_1^U} l_i(w_{A_1^U}) \leq \sum_{i\in A_1^U} l_i(w_V)$이 성립하므로, 다음의 부등식을 얻는다.

$$(w_{A_1^U} - w_V) \sum_{i\in A_1^U} \frac{\partial l_i}{\partial \hat{y}_i}(w_V) \leq 0 \tag{3.5}$$

또한, A_1^U에 대해 다음의 부등식이 성립한다.

$$\sum_{i\in A_1^U} \frac{\partial l_i}{\partial \hat{y}_i}(w_V) \leq 0 \tag{3.6}$$

그 이유는 다음과 같다. 만일 $\sum_{i\in A_1^U} \frac{\partial l_i}{\partial \hat{y}_i}(w_V) > 0$이면, 문제 (2.7)에서 $z^t x$를 최소로 하려면 A_1^U의 모든 원소 i에 대해 $x_i = -1$이고, $A_1^U \subset L$이어야 한다. 왜냐하면, A_1^U가 U에서 가장 작은 최적해를 갖는 블록이기에, 이 집합이 L에 가도 문제 (2.7)의 순서제약을 위배하지 않기 때문이다. 이때, $L \cap U \neq \varnothing$이 되어, 모순이 발생한다. 식 (3.5)와 식(3.6)으로부터, $w_{A_1^U} \geq w_V$가 도출된다. 정의에 의해, $w_{M_1^U} \geq w_{A_1^U}$이 성립하고, 식 (3.2)를 이용하면, $w_{M_1} > w_V$가 성립한다. 마찬가지 논리로 $w_{M_K} < w_V$가 도출되고, 따라서, $w_{M_1} > w_{M_K}$가 성립한다. 그러나 이는 부등식(3.1)과 모순된다. 그러므로 분할되는 블록이 존재하지 않는다. □

이 증명에서 오류는 식(3.5)에서 식(3.6)을 이용하여 $w_{A_1^U} \geq w_V$를 도출하는 부분에 있다. 만약 $\sum_{i\in A_1^U} \frac{\partial l_i}{\partial \hat{y}_i}(w_V) = 0$이 될 때, $w_{A_1^U} = \operatorname{argmin}_z \sum_{i\in A_1^U} l_i(z)$가 유일하지 않다면, $w_{A_1^U}$와 w_V사이의 대소관계에 대해 어떠한 결론도 도출할 수 없다. 후버 함수에서 이런 이유로 인해 반례가 발생하고, 일반적인 볼록함수에 대해서 위 정리가 성립하지 않게 된다. 또한 최종 분할까지 가는 과정에서 나오는 중간 분할 결과물들이 순서 제약을 만족한다는 해당 논문의 Theorem 2도 Theorem 1을 바탕으로 하고 있으므로 오류이다.

이를 해결하기 위해서는 임의의 집합 X에 대해 w_X가 유일해야 한다. 이는 함수 l_i가 모든 i에서 순볼록 함수이면서 강압적 함수, 즉 $\lim_{|x|\to\infty} l_i(x) = \infty$일 때 가능한 조건이다. 순볼록 함수의 합도 순볼록 함수가 되고, 강압적 함수의 합도 강압적 함수가 된다. 그렇기에

l_i가 순볼록이고 강압적 함수이면, 임의의 집합 X에 대해 $\sum_{i \in X} l_i(x)$도 순볼록이고 강압적 함수가 된다. 강압적 함수이고 연속함수이면 함수값이 최솟값이 되는 해가 정의역에 존재한다는 것이 알려져 있고(Lange, 2016), 볼록함수는 연속함수라는 사실을 추가적으로 이용하면, $\sum_{i \in X} l_i(x)$는 최솟값이 되는 해가 정의역에 존재하고, 순볼록 함수라는 사실에 의해 이 최솟값은 유일하다.

결론적으로, l_i가 순볼록 함수이고, 강압적 함수가 되면, 첫번째 정리의 증명에서 발생하는 오류를 방지할 수 있고, GIRP 알고리즘을 사용하여 최적해를 구할 수 있다. 후버 손실함수는 강압적이나 순볼록함수가 아니기에 GIRP 알고리즘에서 현재 분할 시점에서 이전의 분할을 고려할 필요가 없다는 Theorem 1의 결과를 사용할 수 없고, 최적해를 도출하려면 조상 노드의 분할을 고려해야 하는 상황이 발생할 수 있다.

Chapter 4

근위 거리 알고리즘

이 장에서는 근위 거리 알고리즘을 소개하고, 이를 이용해 등위회귀함수 추정 문제를 푸는 과정을 제시하였다.

근위 거리 알고리즘은 유클리드 공간에서 해가 볼록이고 닫힌 제약집합 C에 속해 있는 경우에, 이 제약 하에 손실함수를 최소화하는 해를 구하는 알고리즘이다. 이 때, 해가 C에 속하는 상황은 유클리드 거리함수를 사용해서 나타낼 수 있는데, $dist(\hat{y}, C) = \min_{y \in C} ||\hat{y} - y||$가 0이 되면 벡터 $\hat{y}$가 C에 속하는 것과 동치가 된다. 이런 성질과, 0에서의 미분가능성을 확보하기 위해 거리함수를 오차제곱함수를 사용한다.

$$\min_{\hat{y}} \{l(\hat{y}) + \frac{\lambda}{2}\text{dist}^2(\hat{y}, C)\} = \min_{\hat{y}} \{l(\hat{y}) + \frac{\lambda}{2}||\hat{y} - P_C(\hat{y})||^2\}, \quad \lambda > 0 \tag{4.1}$$

여기서 $l(\hat{y}) = l(\hat{y}, y)$, $P_C(\hat{y}) = \underset{u \in C}{\text{argmin}}||u - \hat{y}||$는 $\hat{y}$의 C공간으로 사영을 의미하며, λ는 적당히 큰 값으로 정한다. C가 닫혀있고, 볼록집합일때, $P_C(x)$는 C에 속하며 유일하다는 것이 알려져 있다. 임의의 벡터 x에 대하여 $||\hat{y} - P_C(\hat{y})|| \leq ||\hat{y} - P_C(x))||$가 성립하므로, MM 알고리즘에서 $f(\hat{y})$의 대리함수 $g(\hat{y}|x)$를 다음과 같이 정의할 수 있다.

$$g(\hat{y}|x) = l(\hat{y}) + \frac{\lambda}{2}||\hat{y} - P_C(x)||^2 \tag{4.2}$$

이때, $f(\hat{y}) = l(\hat{y}) + \frac{\lambda}{2}\text{dist}^2(\hat{y}, C) \leq g(\hat{y}|x)$ $\forall \hat{y}$이 되며, $\hat{y} = x$가 되면 등호가 성립한다. 이를 이용하여, $x = x_m$일때, x_{m+1}를 $g(\hat{y}|x_m)$값을 최소로 하는 해라고 하면, 아래와 같은 부등식이 성립한다.

$$f(x_{m+1}) \leq g(x_{m+1}|x_m) \leq g(x_m|x_m) = f(x_m) \tag{4.3}$$

즉, $g(\hat{y}|x_m)$을 최소로 하는 해를 반복적으로 구함으로써, $f(\hat{y})$값을 점차 작게 할 수 있다.

근위거리 알고리즘의 핵심은 2가지이다. 하나는 m번째 작업에서 $g(\hat{y}|x_m)$을 최소로 하는 해 x_{m+1}를 구하는 작업이고, 다른 하나는 최소화한 해를 C공간으로 사영한 값을 바탕으로 다시 대리함수 $g(\hat{y}|x_{m+1})$을 정의하는 것이다. 그리고, 이런 작업을 반복적으로 하여 식 (4.1)의 우변을 주어진 λ에 대해 최소화한다. 이제 $\lambda \to \infty$으로 두고, 이 최소화 작업을 계속하면, 최종적으로 C에서 $l(\hat{y})$를 최소화하는 해를 구할 수 있게 된다. 만약 l이 볼록함수이면, 그 해는 전역 최소해가 된다. 최소화해야하는 함수를 살펴보면, λ가 작으면 손실함수 l을 최소화하는데 더 초점이 맞춰지고, λ가 무한대로 갈수록 해가 C안에 있도록 하는데 초점이 더 맞춰진다. 그렇기에 초기에 λ를 너무 크게 잡으면 손실함수를 최소화하지 못하면서 C안에만 속하는 해를 구하게 된다. 그렇기에, 실제 알고리즘을 수행시 매번 반복시에 λ 값을 키워가면서 초기에는 해가 손실함수를 최소화하는 값이 되도록 하면서 점차 C안에 속하도록 만들어주는 방법을 사용해야 한다.

위의 과정에서 첫번째 작업인 x_{m+1}을 구하는 작업은 근위사상(proximal map)으로 표현할 수 있다.

$$\mathrm{prox}_l(y) = \underset{x}{\mathrm{argmin}}\{l(x) + \frac{1}{2}||y - x||^2\} \tag{4.4}$$

이때,

$$\mathrm{prox}_{\lambda^{-1}l}(y) = \underset{x}{\mathrm{argmin}}\{\lambda^{-1}l(x) + \frac{1}{2}||y - x||^2\} = \underset{x}{\mathrm{argmin}}\{l(x) + \frac{\lambda}{2}||y - x||^2\} \tag{4.5}$$

와 같고, 이를 이용해 $x_{m+1} = \mathrm{prox}_{\lambda^{-1}l}(y)$로 표현할 수 있다. 즉 첫번째 작업은 함수 $\lambda^{-1}l$의 근위 사상을 찾는 문제가 된다.

두번째 작업은 x_{m+1}을 C공간으로 사영하는 작업인데, 이는 결국 아래의 문제를 푸는 것과 같다.

$$P_C(x_{m+1}) = \underset{y \in C}{\mathrm{argmin}}||y - x_{m+1}||^2 \tag{4.6}$$

C가 순서제약집합일 때, 이 문제는 오차제곱 손실함수 하에서의 등위회귀함수 추정과 동등하므로, 2장의 방법론을 적용할 수 있다.

근위거리 알고리즘은 근위사상과 관련된 방대한 결과들을 사용할 수 있고, l이 미분불가능한 함수여도 적용가능하며, C가 볼록집합이 아니여도 되는 등 다양한 장점이 있다.

Algorithm 4에는 위의 내용을 종합하여, 근위 거리 알고리즘을 이용하여 등위회귀함수를 추정하는 의사 코드를 작성하였다.

Algorithm 4 근위 거리 알고리즘을 이용한 등위회귀함수 추정

Require: 자료가 $(x_1, y_1), ..., (x_n, y_n)$, $y = (y_1, y_2, ..., y_n)$ 주어지고, 부분적으로 순서 제약 공간(C)이 주어진 경우

Require: $f(\hat{y}) = l(\hat{y}) + \frac{\lambda}{2} dist^2(\hat{y}, C)$ 대한 대리함수 $g(\hat{y}|x) = l(\hat{y}) + \frac{\lambda}{2}||\hat{y} - P_C(x)||^2$ 를 구하고, 근위사상(proximal map)를 구한다. $\text{prox}_l(y) = \underset{x}{\text{argmin}}(l(x) + \frac{\lambda}{2}||x - y||^2) = \text{prox}_{\lambda^{-1}l}(y)$, $P_C(x)$는 x를 C공간으로 사영한 값.

초기 해 $\hat{y}_0$를 y로 한다. L은 순서 제약 조건이 담긴 행렬이다. $\hat{y}_{prev}$에 $\hat{y}_0$를 저장한다.ϵ은 작은 값으로 설정한다.

1: **while** 수렴 **do**
2: λ의 값을 증가시킨다.
3: $x_{new} = \text{prox}_{\lambda^{-1}l}(\hat{y}_{prev})$
4: 제곱 손실함수하에서 $\hat{y}_{new} = GIRP(x_{new}, L)$를 구한다.
5: **if** $mean(|\hat{y}_{new} - \hat{y}_{prev}|) \leq \epsilon$ **then**
6: 반복문을 종료한다.
7: **else**
8: $\hat{y}_{prev} = \hat{y}_{new}$로 업데이트 하고, 다시 반복문을 시작한다.
9: **end if**
10: **end while**

Chapter 5

실험결과

본 논문의 실험은 줄리아 0.6.0 버전을 사용하였으며, GIRP에서의 선형 계획법 문제 (2.7)은 Cbc v0.3.2 패키지를 이용해서 계산하였다. 실험의 계산은 1개의 Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz CPU에서의 단일 코어를 사용하였고, 252GB 메모리를 가지는 서버 환경에서 수행하였다.

우선, 2장에서 소개된 2가지 알고리즘의 이론상의 시간 복잡도를 실제 알고리즘에 적용하여 확인해보았다. 복잡도 계산을 위해서 Luss and Rosset (2014)에서 사용한 포아송 분포에서 추출된 표본들을 이용해 시뮬레이션을 하였다. 다음으로는, 절대오차 손실함수와 후버 손실함수에 대하여 근위거리 알고리즘을 적용하였다. 실제 구현 과정에서는 파라메트릭 탐색 알고리즘의 속도가 느려서 GIRP 알고리즘만 적용하였고, 근위사상의 경우 결과물만 제시하고, 결과물의 구체적인 도출과정은 부록에 담았다.

5.1 기존 알고리즘 비교

파라메트릭 탐색 알고리즘의 경우, 이론상으로 $O(mn \log n^2/m)$의 시간 복잡도를 가지는데, 자료의 크기가 커질수록, 속도가 많이 저하되는 현상이 보였다. 이밖에도, 기본적으로 이 알고리즘은 추정값이 정수라는 제약조건하에, 특정 범위 안에 있다는 제약을 설정하는데 이 역시 실제 추정값의 많은 차이를 발생시켰다. 우선, 이 알고리즘의 속도 향상은 추정값이 될 수 있는 모든 후보값을 다 탐색하는 것이 아니라, 현재 탐색하는 두 값의 중간값을 탐색하는 방식으로 탐색하는 후보값들의 수를 줄임으로써 달성된다. 하지만, 현재 탐색하는

값이 앞에서 언급된 중단점이 되는지 여부를 판단하는 부분에 대해 Gallo et al. (1989)에서 자세히 다뤄지지 않고 있다. 중단점의 특성상 주위 점들의 값들을 살펴봐야할 필요성이 있어보이나, 이렇게 하면 이 알고리즘의 속도가 느려진다. 또한, 이 알고리즘은 추정값이 특정 범위 안에 있음을 전제함을 입력값으로 주어진 상태에서 최적해를 구하는데, 범위가 바뀔 때마다 결과물이 크게 달라지는 경향성을 보인다. 마지막으로, 이 알고리즘은 구현에 있어서 최근에 나온 GIRP 방법에 비해 복잡한 편이다.

GIRP 알고리즘의 경우, 이론상으로 $O(n^3)$의 시간 복잡도를 가지는데, 실제 포아송 분포에서 추출된 표본들을 이용해 시뮬레이션 해본 결과,전체 자료를 다 탐색하지 않고, 매 단계별로 시행되는 선형 계획법 솔버 작동에 걸리는 시간이 단축되는 데에서 계산 시간이 크게 감소한 것으로 보인다. 이후 시뮬레이션 자료와 실제 자료에 적용에는 GIRP 알고리즘만 사용하였으며, λ값은 1.1에서 1.2 사이의 값을 주로 사용하였고 포아송 손실함수의 경우에는 20을 사용하였다.

Table 5.1: 두 알고리즘의 구현 속도 비교(단위 : 초)

자료크기	**파라메트릭 탐색**	**GIRP**
100	1.769	0.031
200	6.286	0.073
300	59.91	0.082

5.2 근위거리 알고리즘 구현

Luss and Rosset (2014)에 제시된 2 종의 시뮬레이션 예제에 대하여 GIRP를 적용한 결과물과 GIRP를 사용한 근위거리 알고리즘으로 얻은 결과물이 유사함을 보임으로써 실제 근위거리 알고리즘이 잘 작동함을 보이고자 한다. 각각의 시뮬레이션에서 학습에는 3000개의 자료를 사용하고, 테스트에는 1000개의 자료를 사용하였다. 그리고 10번의 서로 다른 시뮬레이션을 통하여, 나온 테스트 자료의 예측 에러값들의 평균과 표준편차를 계산한 결과물을 제시하였다.

먼저 표 5.3은 균등분포에서 생성된 자료를 바탕으로 순서제약 조건을 만들고, 포아송분포에서 생성된 자료를 이용하여 얻은 결과물이다. 0에서 10 사이의 값을 갖는 균등분포에서 차원의 크기 d 만큼의 자료 $x_i = (x_{i1}, ..., x_{id})$를 뽑은 후, $\Pi_j \sqrt{x_{ij}}$를 포아송분포의 평균으로 하는 분포에서 자료 1개를 뽑아 y_i라고 한다. 그리고 x_i를 통해 순서 제약 조건을 만든다.
표 5.5는 균등분포에서 생성된 자료를 바탕으로 순서제약 조건을 만들고, 정규분포에서 생성된 자료를 이용하여 얻은 결과물이다. 0에서 3 사이의 값을 갖는 균등분포에서 차원의 크기 d 만큼의 자료 $x_i = (x_{i1}, ..., x_{id})$를 뽑은 후, $\Pi_j x_{ij}$를 평균으로 하고, d^2을 분산으로 하는 정규분포에서 자료 1개를 뽑아 y_i라고 한다. 그리고 x_i를 통해 순서 제약 조건을 만든다.
GIRP 알고리즘의 경우, 학습오차를 가장 최소로 하는 마지막 모델을 이용하여, 테스트 자료의 예측값을 도출하였다. 결과물을 살펴보면, 포아송 손실함수에 대해 GIRP 알고리즘을 적용한 결과물과 근위거리 알고리즘을 적용한 결과물이 같고, 오차제곱 손실함수의 경우에서도 같다. 절대오차 손실함수는 GIRP 알고리즘으로 구현이 불가하지만, 근위거리 알고리즘으로 계산이 되기에 추가했다. 3장에서 살펴봤듯이, 후버 손실함수는 GIRP 알고리즘으로 항상 최적해를 구할 수 없지만, 근위거리 알고리즘으로 계산이 가능하다. 후버 손실함수의 경우 δ 값을 0.9로 설정하였다. 이를 통하여 본 논문에서 구현한 근위거리 알고리즘이 정상적으로 작동하고 있음을 확인할 수 있다.

Table 5.2: 손실함수 정리

손실함수 종류	**식**
포아송 손실함수	$l_i(\hat{y}_i) = \hat{y}_i - y_i \ln \hat{y}_i$
오차제곱 손실함수	$l_i(\hat{y}_i) = (\hat{y}_i - y_i)^2$
절대오차 손실함수	$l_i(\hat{y}_i) = \lvert\hat{y}_i - y_i\rvert$
후버 손실함수	$l_i(\hat{y}_i) = \begin{cases} \frac{(\hat{y}_i - y_i)^2}{2} & \lvert\hat{y}_i - y_i\rvert < \delta \\ \delta(\lvert\hat{y}_i - y_i\rvert - \frac{\delta}{2}) & \lvert\hat{y}_i - y_i\rvert \geq \delta \end{cases}$

Table 5.3: 포아송 분포 시뮬레이션 자료에 적용한 사례 : 평균(표준편차)

Model 1 : $y_i \sim \text{Poisson}(\Pi_j \sqrt{x_{ij}})$ with $x_{ij} \sim U[0, 10]$

손실함수	방법	차원				
		2	3	4	5	6
포아송	GIRP	2057.58 (39.79)	2474.03 (35.53)	3134.40 (58.00)	6495.45 (404.37)	28765.13 (1286.10)
	근위거리	2057.58 (39.79)	2474.03 (35.53)	3134.40 (58.00)	6495.45 (404.37)	28765.13 (1286.10)
오차제곱	GIRP	4.529 (0.266)	10.75 (0.78)	39.03 (3.42)	430.37 (71.15)	6659.08 (625.50)
	근위거리	4.529 (0.266)	10.75 (0.78)	39.03 (3.42)	430.37 (71.15)	6659.08 (625.50)
절대오차	근위거리	1.621 (0.06)	2.44 (0.08)	4.35 (0.15)	13.50 (0.617)	57.60 (1.68)
후버	근위거리	1.108 (0.05)	1.83 (0.07)	3.54 (0.14)	11.76 (0.55)	51.43 (1.51)

Table 5.4: 정규분포 시뮬레이션 자료에 적용한 사례 : 평균(표준편차)

Model 2 : $y_i = (\Pi_j x_{ij}) + N(0, d^2)$ with $x_{ij} \sim U[0, 3]$

손실함수	방법	차원				
		2	3	4	5	6
오차제곱	GIRP	15.98 (0.62)	83.72 (2.81)	285.67 (9.26)	773.14 (23.71)	1666.77 (68.84)
	근위거리	15.98 (0.62)	83.72 (2.81)	285.67 (9.26)	773.14 (23.71)	1666.77 (68.84)
절대오차	근위거리	3.19 (0.05)	7.28 (0.13)	13.50 (0.21)	22.19 (0.27)	32.50 (0.75)
후버	근위거리	2.49 (0.05)	6.16 (0.12)	11.75 (0.19)	19.57 (0.24)	28.84 (0.68)

5.3 실제 자료 적용 사례

5.3.1 이상점 보정하기

Luss and Rosset (2014)에서 사용한 2010년 자동차 연비 자료(Frank and Asuncion, 2010)는 392개의 자료와 7개의 독립변수가 사용되며, 종속변수는 연비(mpg)이다. 이때, 7개의 변수를 다 사용하여 다중회귀분석을 실시하고, 그때 도출된 각각의 변수별 계수값의 절댓값이 큰 순서대로 변수를 다시 나열한다. 계수값이 음수인 경우에는 해당 독립변수가

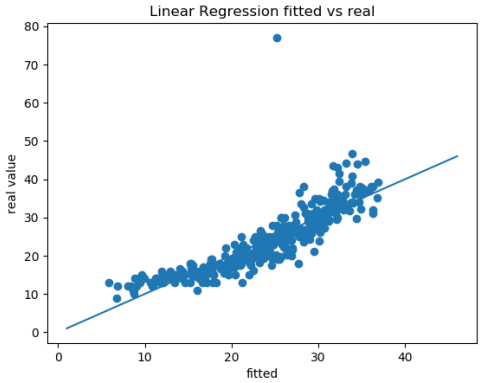
감소할 때 연비가 증가하는 관계가 됨으로, 이를 본 논문에서처럼 독립변수가 증가할 때, 종속변수가 증가하는 제약 형태로 바꾸기 위해, 해당 독립변수의 값에 -1을 곱한 값을 사용하여 순서제약을 준다. 이러한 전처리 작업 후, 변수를 차례대로 하나씩 증가시켜 가면서 검증자료에 대한 평균제곱오차(mean squared error, 이하 MSE) 값을 정리하였다. 검증자료는 10-fold cross-validation 방법으로 추출하였다. 그 다음 학습자료에서 임의의 값을 하나 뽑아서 77으로 이상점을 만들어주고, 새로 생긴 이상점의 회귀함수 추정값이 주변의 5개 값보다 작다는 제약조건을 추가한다.

아래 테이블 결과물을 살펴보면, 변수가 1개에서 점차 증가하면서 선형회귀분석 방법보다 등위회귀함수에서의 MSE가 더 작아짐을 알 수 있다. 하지만 변수가 더 많아질수록, 모든 차원의 값이 다 커야 순서 제약이 성립하므로, 순서 제약 조건이 적어지고, 이로 인해 학습자료에서의 과적합 현상이 일어난다. 왜냐하면 아무런 제약이 없을 때, 가장 손실함수를 작게 해주는 값은 자료값 그 자체를 추정값으로 하는 것이기 때문이다. 이로 인해 선형회귀분석 방식은 변수가 늘어날수록 MSE가 계속 감소하지만, 등위회귀분석 방식은 변수가 많아질수록 MSE 값이 다시 커지는 현상을 보인다. 여기서 후버 손실함수의 δ값은 1.65로 하였다.

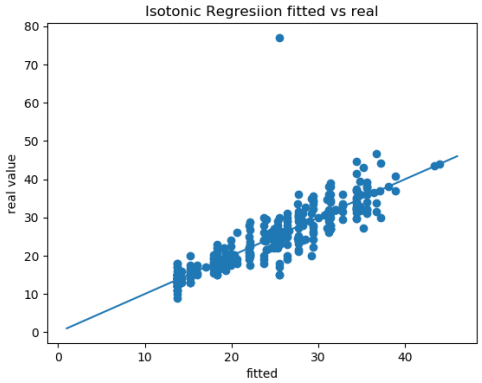
Table 5.5: 자동차 자료 이상점 추가시 검증자료 MSE

변수개수	**선형회귀**	**Isotonic** L_2	**Isotonic huber**	**Isotonic** L_1
1	41.64	44.28	44.16	44.07
2	27.28	25.00	24.89	24.87
3	16.26	14.08	**13.90**	14.03
4	16.58	14.27	**14.11**	14.21
5	15.99	16.85	16.44	16.91
6	15.82	17.49	16.44	16.67
7	**12.89**	33.45	33.10	33.04

등위회귀함수에서는 후버 손실함수를 사용했을 때, 오차제곱이나 절대오차 손실함수의 경우보다 MSE가 상대적으로 낮았다. 그리고 등위회귀함수에서는 변수 3개를 쓰고, 후버



(a) 변수 7개의 중회귀함수
x축은 추정값, y축은 실제값



(b) 변수 4개 등위회귀함수(후버 손실함수)
x축은 추정값, y축은 실제값

Figure 5.1: 이상점 있을 때의 학습자료 예측결과

손실함수를 사용한 경우 MSE가 가장 낮게 나왔다. 한편, 7개 변수를 다 쓴 회귀분석에서의 MSE값이 전체적으로 낮았는데, 실제 그림을 그려보면 추정값들이 이상점에 의해 영향을 받아 전체적으로 값들이 과대평가되는 문제점이 있는 것으로 나온다. 아래 그림은 실제 mpg값이 25.5인 값을 77로 만들었을 때, 중회귀분석과 4개 변수하의 등위회귀분석(후버 손실함수)의 학습자료 추정값과 실제값을 산점도로 나타낸 그림이다. 중회귀분석 결과물을 보면 이상점을 제외한 값들이 대체로 기울기가 1인 선 아래쪽에 위치해있다. 이는 추정값이 실제값보다 큼을 의미하고, 과대평가되는 문제를 보여준다. 반면에 등위회귀분석 결과물은 대체로 추정값이 기울기가 1인 선 주변으로 분포되어 있어, 추정값과 실제값을 잘 반영하고 있으며, 이상점의 영향을 받지 않는다.

Chapter 6

결론

근위 거리 알고리즘을 이용한 방법은 근위사상과 오차제곱 손실함수 하에서 등위회귀함수를 추정하는 기존 알고리즘을 결합하여, 등위회귀함수에 사용 가능한 손실함수의 범위를 확장시켰다. 특히, 부분적으로 순서 제약 조건이 주어진 상황에서, 기존에 적용하기 어려웠던 절댓값 손실함수나 힌지 손실함수를 사용할 수 있게 되었고, 자료 각각에 대한 최적해를 구하는 방식으로 구현이 되기 때문에, 후버 손실함수에 GIRP 알고리즘을 적용했을 때 생기는 문제점도 해결할 수 있다. 즉, 다양한 손실함수 하에서 실수해를 구할 수 있게 되었다. 이를 통해, 변수가 적을 때, 순서제약이 있는 회귀선을 추정하는데 기존의 선형회귀분석 방식보다 MSE값이 작았으며, 이상점이 존재할 때에도 회귀함수가 이상점의 영향을 받지 않는 점을 확인할 수 있었다.

본 논문에서는 근위사상을 적용한 방법론과 일부 예제를 소개했는데, 소개하지 않은 손실함수에 대해서도 근위사상이 쉽게 계산 가능하면 적용 가능하다. 또한, 등위회귀함수의 손실함수 범위를 확장시킬 수 있는 다른 방법이 제안된다면, 속도와 안정성 측면에서 어떤 방법이 더 효율적인지를 검증해볼 필요가 있다. 한편, 이 알고리즘은 병렬처리 방식을 이용하여, 큰 자료를 적용할 때 빠른 속도로 작동하도록 개선될 여지가 있다.

부록

Example 1 (포아송 손실함수)**.** $f(\hat{y}) = \sum_{i=1}^{n}(\hat{y}_i - y_i \ln(\hat{y}_i))$일때, y_i에서의 근위사상은 $(\lambda y_i - 1) + \sqrt{(\lambda y_i - 1)^2 + 4\lambda b_i}/(2\lambda)$이 되고, b_i 는 i번째 자료 값을 의미한다.

Proof. $\hat{y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_n)$에서, 벡터의 미분은 각각의 원소로 미분하기에, 임의의 원소에 대한 결과물만 살피면 된다. 이 때, 최적해인 $\hat{y_i}^*$는 $0 = \frac{1}{\lambda}(1 - \frac{b_i}{\hat{y}_i}) + (\hat{y}_i - y_i)$을 만족해야 한다. 이를 정리하면 $\lambda\hat{y}^2 + (1 - \lambda y_i)\hat{y}_i - b_i = 0$이 되고, 근의 공식에 의해 이차방정식의 해가 2개가 나오는데, 그 중 하나는 음수이고, 하나는 양수인데, 포아송 분포는 양수의 값만 가능하므로, 둘 중에 양수만을 택한다. □

Example 2 (오차제곱 손실함수)**.** $f(\hat{y}_i) = \sum_{i=1}^{n}(\hat{y}_i - b_i)^2$일때, y_i에서의 근위사상은 $\hat{y}_i = (2b_i + \lambda y_i)/(2 + \lambda)$이 되고, b_i 는 i번째 자료 값을 의미한다.

Proof. $\hat{y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_n)$에서, 벡터의 미분은 각각의 원소로 미분하기에, 임의의 원소에 대한 결과물만 살피면 된다. 이 때, $\hat{y}_i$의 최적해인 $\hat{y_i}^*$는 $0 \in \frac{1}{\lambda}2(\hat{y} - b_i) + (\hat{y}_i - y_i)$이라는 조건식을 만족해야하는데, 이를 정리하면, 위의 결과물이 도출된다. □

Example 3 (절대오차 손실함수)**.** $f(\hat{y}) = \sum_{i=1}^{n} |\hat{y}_i - b_i|$ 일때, y_i에서의 근위탐색은 $\hat{y}_i^* = \text{sgn}(y_i - b_i)\max(0, |y_i - b_i| - \frac{1}{\lambda}) + b_i$이고, 여기서 $\hat{y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_n)$이고, b_i 는 i번째 자료 값을 의미한다.

Proof. $\hat{y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_n)$에서, 벡터의 미분은 각각의 원소로 미분하기에, 임의의 원소에 대한 결과물만 살피면 된다. 그래서 $\hat{y}_i$의 최적값을 구하는 과정만 증명하면 충분하다. $\hat{y}_i$의 최적해인 $\hat{y_i}^*$는 $0 \in \frac{1}{\lambda}\partial f(\hat{y}_i) + (\hat{y}_i - y_i)$이라는 조건식을 만족해야하는데, 이때 $\partial f(\hat{y}_i)$는 $f(\hat{y}_i)$의 보조미분(subdifferential)을 의미한다. 이때, $\hat{y_i}^*$가 b_i와 다를 때와 같을 때, 조건식은 다르게 표현되기 때문에, 이 2가지 경우를 각각 살펴보고자 한다.

먼저, 첫번째 경우로 $\hat{y_i}^* \neq b_i$인 경우, $\partial f(\hat{y_i}^*) = \text{sgn}(\hat{y_i}^* - b_i)$로 표현되고,그러므로 $0 = \frac{1}{\lambda}\text{sgn}(\hat{y_i}^* - b_i) + (\hat{y_i}^* - y_i)$가 되어, $\hat{y_i}^* = y_i - \frac{1}{\lambda}\text{sgn}(\hat{y_i}^* - b_i)$가 된다. 이때, 아래와 같은 이유로 $\text{sgn}(\hat{y_i}^* - b_i)$는 $\text{sgn}(y_i - b_i)$와 같아져서 이 경우에는 $\hat{y_i}^* = y_i - \frac{1}{\lambda}\text{sgn}(y_i - b_i)$가 된다.

$$(|\hat{y_i}^* - b_i| > 0 \Leftrightarrow |y_i - b_i| > \frac{1}{\lambda}) \Leftrightarrow \begin{cases} \hat{y_i}^* > b_i \Leftrightarrow \hat{y_i}^* = y_i - \frac{1}{\lambda} > b_i \Leftrightarrow y_i - b_i > \frac{1}{\lambda} \\ \hat{y_i}^* < b_i \Leftrightarrow \hat{y_i}^* = y_i + \frac{1}{\lambda} < b_i \Leftrightarrow y_i - b_i < -\frac{1}{\lambda} \end{cases}$$

두번째 경우는 $\hat{y_i}^* = b_i$인 경우인데, 이때, $0 \in \frac{1}{\lambda}[-1, 1] + \hat{y_i}^* - y_i$가 성립하고, 여기서 $[-1, 1]$은 -1에서 1사이의 값을 의미한다. 그러므로 $0 \in \frac{1}{\lambda}[-1, 1] + b_i - y_i$가 되고, 이는 $y_i - b_i \in \frac{1}{\lambda}[-1, 1]$와 동치가 되고 이는 다시 $|y_i - b_i| \leq \frac{1}{\lambda}$과 동치가 된다.

위의 결과를 종합하면, 다음의 결과물이 나온다.

$$\hat{y_i}^* = \begin{cases} b_i & |y_i - b_i| \leq \frac{1}{\lambda} \\ y_i - \frac{1}{\lambda}\text{sgn}(y_i - b_i) & |y_i - b_i| > \frac{1}{\lambda} \end{cases}$$

이를 한줄로 정리하면 $y_i^* = \text{sgn}(y_i - b_i)\max(0, |y_i - b_i| - \frac{1}{\lambda}) + b_i$가 된다. □

Example 4 (후버 손실함수)**.**

$$f(\hat{y}_i) = \begin{cases} \frac{(\hat{y}_i - b_i)^2}{2} & |\hat{y}_i - y_i| < \delta \\ \delta(|\hat{y}_i - | - \frac{\delta}{2}) & |\hat{y}_i - y_i| \geq \delta \end{cases}$$

y_i에서의 근위사상은

$$\hat{y}_i^* = \begin{cases} (y_i\lambda + b_i)/(\lambda + 1) & b_i - (1 + \frac{1}{\lambda})\delta \leq y_i \leq b_i + (1 + \frac{1}{\lambda})\delta \\ (y_i - \frac{1}{\lambda}\delta) & y_i \geq b_i + (1 + \frac{1}{\lambda})\delta \\ (y_i + \frac{1}{\lambda}\delta) & y_i \leq b_i - (1 + \frac{1}{\lambda})\delta \end{cases}$$

이 되고, b_i 는 i번째 자료 값을 의미한다.

Proof. 여기서도 i번째 최적해인 $\hat{y}_i^*$는 $0 \in \frac{1}{\lambda}\partial f(\hat{y}_i^*) + \hat{y}_i^* - y_i$를 만족해야한다.
먼저, $|\hat{y}_i^* - b_i| < \delta$ 인 경우 $0 = \frac{1}{\lambda}(\hat{y}_i^* - b_i) + \hat{y}_i^* - y_i$이 성립하고, 이를 정리하면 첫번째 경우가 나온다. 이때, 이를 대입해 다시 $|\hat{y}_i^* - b_i| < \delta$를 y_i에 대한 부등식으로 정리하면, $b_i - (1 + \frac{1}{\lambda})\delta \leq y_i \leq b_i + (1 + \frac{1}{\lambda})\delta$가 나온다.

두번째로, $\hat{y}_i^* - b_i \geq \delta$인 경우, $0 = \frac{1}{\lambda}\delta + \hat{y}_i^* - y_i$가 성립하고, 이를 정리하면, $\hat{y}_i^* = y_i - \frac{1}{\lambda}\delta$가 되고, 이를 다시 두번째 경우에 적용하면, $y_i \geq b_i + (1 + \frac{1}{\lambda})\delta$가 나온다.
마찬가지로, 세번째 경우는 두번째 경우와 마찬가지 방식으로 적용하면 도출된다. □

Example 5 (힌지 손실함수)**.** $f(\hat{y}_i) = \max(0, 1 - b_i\hat{y}_i)$일때, y_i에서의 근위사상은

$$\hat{y}_i^* = \begin{cases} y_i & |y_i| > 1 \\ 1 & 1 - \frac{1}{\lambda} \leq y_i \leq 1 \\ -1 & -1 + \frac{1}{\lambda} \leq y_i \leq -1 \\ y_i + \frac{b_i}{\lambda} & -1 + \frac{1}{\lambda} \leq y_i \leq 1 - \frac{1}{\lambda} \end{cases}$$

이 되고, b_i 는 i번째 자료 값을 의미한다.

Proof. 여기서도, 일변수에 대한 결과물만 도출하면, 다변수에 대해 일반화를 할 수 있기 때문에, 일변수인 경우만 증명한다. $\hat{y}$의 최적값을 구하는 과정만 증명하면 충분하다. $\hat{y}$의 최적해인 $\hat{y}^*$는 $0 \in \frac{1}{\lambda}\partial f(\hat{y}) + \hat{y} - y$를 만족해야 하는데, 이때 $\hat{y}$는 다음의 3가지 경우 중 하나에 속한다. b는 자료 값을 의미하며 1이거나 -1의 값을 갖는다.

먼저, $1 - b\hat{y}^* < 0$인 경우, $f(\hat{y}^*) = 0$이 되고, 그 결과 $0 = \hat{y}^* - y$가 되어 $\hat{y}^* = y$가 된다. 이 때, $b = 1$인 경우, $1 < \hat{y}^*$가 성립하고, $b = -1$인 경우 $\hat{y}^* < -1$이 성립하기 때문에 $1 - b\hat{y}^* < 0$은 $|\hat{y}^*| > 1$과 동치가 되고, 이는 다시 $|y| > 1$과 동치가 된다.
두번째로, $1 - b\hat{y}^* = 0$인 경우,$\hat{y}^*$는 $0 \in \frac{1}{\lambda}[-b, 0] + \hat{y} - y$를 만족해야 하는데, 다음과 같은 경우로 살펴볼 수 있다.

$$\begin{cases} b = 1 \Leftrightarrow \hat{y_i}^* = 1 \Leftrightarrow y \in 1 + \frac{1}{\lambda}[-1, 0] \Leftrightarrow 1 - \frac{1}{\lambda} \leq y \leq 1 \\ b = -1 \Leftrightarrow \hat{y_i}^* = -1 \Leftrightarrow y \in -1 + \frac{1}{\lambda}[0, 1] \Leftrightarrow -1 \leq y \leq -1 + \frac{1}{\lambda} \end{cases}$$

마지막으로, $1 - b\hat{y}^* > 0$인 경우, $\hat{y}^*$는 $0 = -\frac{b}{\lambda} + \hat{y} - y$를 만족해야 하는데 두번째 경우와 유사한 방식으로 다음과 같이 정리할 수 있다.

$$\begin{cases} b = 1 \Leftrightarrow \hat{y_i}^* < 1 \Leftrightarrow y + \frac{1}{\lambda} < 1 \\ b = -1 \Leftrightarrow \hat{y_i}^* > -1 \Leftrightarrow y - \frac{1}{\lambda} > -1 \end{cases}$$

결론적으로 $1 - b\hat{y}^* > 0$인 경우는 $-1 + \frac{1}{\lambda} < y < \frac{1}{\lambda}$와 동치가 된다.

□

Bibliography

Bezanson, J., S. Karpinski, V. B. Shah, and A. Edelman (2012). Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*.

Boyd, S. and L. Vandenberghe (2004). *Convex optimization*. Cambridge university press.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein (2009). *Introduction to algorithms*. MIT press.

Frank, A. and A. Asuncion (2010). UCI machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california. *School of information and computer science 213*, 2–2.

Gallo, G., M. D. Grigoriadis, and R. E. Tarjan (1989). A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing 18*(1), 30–55.

Hochbaum, D. S. and M. Queyranne (2003). Minimizing a convex cost closure set. *SIAM Journal on Discrete Mathematics 16*(2), 192–207.

Huber, P. J. et al. (1964). Robust estimation of a location parameter. *The annals of mathematical statistics 35*(1), 73–101.

Lange, K. (2016). *MM Optimization Algorithms*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Luss, R. and S. Rosset (2014). Generalized isotonic regression. *Journal of Computational and Graphical Statistics 23*(1), 192–210.

Luss, R., S. Rosset, M. Shahar, et al. (2012). Efficient regularized isotonic regression with application to gene–gene interaction search. *The Annals of Applied Statistics 6*(1), 253–283.

Murty, K. (1983). *Linear Programming*. Wiley.

Spouge, J., H. Wan, and W. Wilbur (2003). Least squares isotonic regression in two dimensions. *Journal of Optimization Theory and Applications 117*(3), 585–605.

Stout, Q. F. (2013). Isotonic regression via partitioning. *Algorithmica 66*(1), 93–112.

Wahba, G. (1999). Support vector machines, reproducing kernel hilbert spaces, and randomized gacv. In *Advances in kernel methods*, pp. 69–88. MIT Press.

Abstract

GIHAN JUNG
Department of Statistics
The Graduate School
Seoul Natinoal University

This paper deals with algorithms for solving the problem of estimating isotonic regression functions under partial order constraints. When the cost function is an error square function, this problem becomes a quadratic programming problem, and an efficient algorithm using a recursive division method is known. Although generalization has been proposed for the case where the cost function is an arbitrary differentiable convex function, there is an error in applying huber loss function, and it cannot applied to cost functions that can not be partially differentiated, such as an absolute loss function and a hinge loss function. In this paper, I show the problems of the generalized-isotonic-recursive-partitioning(GIRP) algorithm under the huber loss function, explain the cause of the problem, and present a range of applicable function space.Then, I propose an algorithm that can be applied to the loss functions that can not be solved by previous algorithms, through the proximity distance algorithm, which is a kind of the MM algorithm. In addition, the theoretical speeds of the two algorithms to be used in projecting the optimal solution of the upper bound function into a sequence constrained set in the implementation of the proximal distance algorithm are compared and the actual performance is presented through simulation. Finally, I demonstrate the results for estimating isotonic regression function on real data, under an absolute loss function and a huber loss function by using the proximal distance algorithm.

Keywords: Isotonic regression,GIRP algorithm,parametric search mincut algorithm,proximal distance algorithm,MM algorithm
Student Number: 2017-26998