2019-09-25

computational Statistics

HW#2

192STG11 우나영

computational Statistics

HW#2

1. Problem

과제 1 에서 bisection, newton, 그리고 secant 알고리즘이 단봉 함수에서 어떻게 optima 에 수렴하는지 실현해 보았다. 이번 과제 2 에서는 이러한 방법론들이 optima 가 복수인 다봉 함수 일때 어떻게 수렴하는지 실현해 볼 것이다. 다봉 함수는 optima 가 한 개 이상이기 때문에 전체 함수의 optima 인 global optima 와 구간별 local optima 로 나뉜다. 최적의 optimization 결과는 global optima 에 수렴하는 것이다. 하지만 과제 1 의 단봉 함수 예제에서 알고리즘들이 초기값에 따라 optima 에 수렴 여부와 수렴의 속도가 달라졌듯이 다봉 함수에서는 알고리즘들이 초기값에 따라 더 다양한 optima 의 수렴양상을 보일 것으로 예상된다.

앞서 배운 세가지의 알고리즘을 통해 optimization 의 문제들을 해결할 수 있다고 생각하는 것은 큰 오산이다. 세상의 수 많은 optimization 문제들 앞에서 인간은 겸손해지기 마련이다. 그 대표적인 optimization 문제가 바로 경우의 수이다. 예를 들어 Travelling Salesman Problem(TSP)에서 문제의 핵심은 출발점에서 가야할 도시를 모두 거쳐 출발점으로 돌아오는 단거리의 방법을 찾는 것이다. 이와 같은 경우의 수 문제들을 완벽하게 해결하는 방법은 모든 경우를 따져 보는 것이다. 하지만 TSP 의 문제에서 도시의 수가 하나만 늘어나도 경우의 수는 기하급수적으로 많아진다. 따라서 이러한 경우의 수 문제에서 global optima 를 찾기 위해 모든 경우의 수를 따지는 것은 비효율적이기때문에 경쟁력 있는 local optima 를 제한된 시간내에 찾는 방법이 필요하다.

그 대안 중 하나로서 local search 가 있다. Local search 는 제한된 search 범위 안에서 현재의 후보 솔루션$\theta^{(t)}$보다 더 나은 다음의 후보 솔루션$\theta^{(t+1)}$으로 반복적으로 이동(iterative procedure) 하여 local optima 에 도달하면 멈춘다. 이때 이러한 이동을 step 또는 move 라고 부른다. Local search 방법은 비록 모든 가능한 경우의 수를 고려하지 않아 uncompetitive 한 optima 로 수렴할 수 있다는 약점이 있지만 제한된 search 범위안에서 가능한 빠른 시간 안에 경쟁력 있는 local optima 를 찾는 다는 점에서 강점이 있다.

본 과제에서는 경우의 수 문제 중 하나인 회귀 분석의 변수 선택 문제를 여러가지 local search 방법론으로 최적화해볼 것이다. 본 과제에서 다룰 데이터는 Baseball Players' Salaries 로 아래와 같다.

y	salary	연봉	x14	arbitration	연봉 조정
x1	average	타율	x15	runsperso	득점/삼진
x2	obp	출루율	x16	hitsperso	안타/삼진
x3	runs	득점	x17	hrsperso	홈런/삼진
x4	hits	안타	x18	rbisperso	타점/삼진
x5	doubles	2 루타	x19	walksperso	4 구/삼진
x6	triples	3 루타	x20	obpererror	출루율/실책
x7	homeruns	홈런	x21	runspererror	득점/실책
x8	rbis	타점	x22	hitspererror	안타/실책
x9	walks	4 구	x23	hrspererror	홈런/실책
x10	sos	삼진	x24	soserrors	삼진*실책
x11	sbs	도루	x25	sbsobp	도루*출루율
x12	errors	실책	x26	sbsruns	도루*득점
x13	freeagent	자유 계약 선수	x27	sbshits	도루*안타

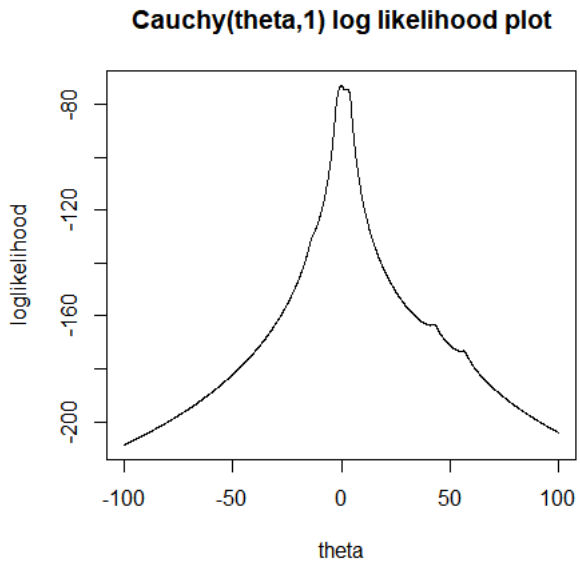
Baseball Players' Salaries 는 투수를 제외한 377 명의 야구선수들의 1991 년 경기 실적과 1992 년 그들의 연봉에 대한 데이터이다. 데이터는 총 27 개의 경기 실적과 관련된 설명 변수와 1 개의 연봉이라는 반응변수로 이뤄졌다. 위의 데이터를 통해 이전 시즌의 경기 실적을 바탕으로 야구선수의 연봉을 예측하는 통계 모델링을 할 수 있다. 본 과제에서는 연봉에 log 를 취한 값을 반응 변수로 둘 것이다. 가능한 모든 회귀 모델의 수는 2^{27}=134,217,728 개이지만 다양한 local search 방법론을 사용하여 가장 optimal 한 선형 모델 즉 best subset 을 가진 모델을 구할 것이다.

2. Result

연습문제

ex 2.1

(a) newton Cauchy(theta, 1)

# 2.1-(a) #	
$= \frac{1}{\pi\gamma(1+\left(\frac{x-\theta}{\gamma}\right)^2)} \sim \text{Cauchy}(\theta, 1)$ $= \sum_{i=1}^{n} logf(x_i;\theta,1)$ $= -\sum_{i=1}^{n} log(1+(x-\theta)^2)$	

method : newton(소수점 네번째 자리에서 반올림)

initial value	-11	-1	0	1.5	4	4.7	7	8	38	5.106
θ^*	X	-0.192	-0.192	1.714	2.817	-0.192	41.041	X	42.795	54.877
$\ell(\theta^*)$	X	-72.916	-72.916	-74.642	-74.360	-72.916	-163.608	X	-163.313	-173.335
iteration	X	1001	97	7	6	1001	10	X	6	10
time	X	0.3793	0.0386	0.005	0.0025	0.3896	0.0046	X	0.0025	0.0078

초기값이 -11 과 8 일 때는 값이 나오지 않는다. 다른 초기값들은 각자의 인접한 local optima 를 찾는다. global optima 는 -0.192 로 초기값이 0 과 4.7 일 때 수렴에 성공한다. 데이터의 mean 값인 5.106 이 초기값일 때 θ^*의 값은 54.877 로 수렴하여 global optima 와 동 떨어진 곳에서 수렴한다. 따라서 이를 통해 global optima 를 찾는 것이 목표라면 newton 초기값으로 데이터의 평균이 좋은 수치가 될 수 없음을 알 수 있다.

(b) bisection

initial value	[-1, 1]	[-1, 5]	[-102, -101]
θ^*	-0.192	2.817	-101.999
$\ell(\theta^*)$	-72.916	-74.360	-209.497
iteration	34	35	33
time	0.0128	0.0172	0.013

초기값이 [-1,1] 일 때 반복횟수는 34 회이며 θ^*은 -0.192 이고 mle 에서 log likelihood 의 값은 -72.916 이며 런타임은 0.0128 이다. 초기값이 [-1,5] 일 때는 반복횟수는 35 회이며 θ^*은 2.817 이고 mle 에서 log likelihood 의 값은 -74.360 이며 런타임은 0.0172 이다. Bisection 이 global optima 를 찾지 못하는 경우는 위의 그래프에서 보았을 때 global optima 가 포함되지 않은 구간을 초기값으로 선택했을 경우이다. 하지만 bisection 은 global optima 는 아니더라도 구간 안에 존재하는 하나의 local optima 를 무조건 찾는다. 따라서 local minimum 도 없을 것으로 예상되는 [-102, -101]을 초기값으로 대입했을 때 반복횟수는 33 회미여 θ^*은 -101.999 이고 θ^*에서 log likelihood 의 값은 -209.497 이며 런타임은 0.013 이다. 위의 값은 local opitma 가 없기 때문에 결국 -102 에 수렴한 것으로 보인다.

(d) secant

initial value	(-2, -1)	(-3, 3)	(-11, -10)	(3, 5)	(4.5, 5)	(6.5, 7.1)	(7, 9)	(37, 39)
θ^*	-0.192	2.817	X	2.817	-0.192	X	X	42.795
$\ell(\theta^*)$	-72.916	-74.360	X	-74.360	-72.916	X	X	-163.313
iteration	25	8	X	7	11	X	X	10
time	0.0152	0.0059	X	0.0052	0.0065	X	X	0.0081

(-2, -1)을 대입했을 때 global optima 가 나오고 (-3, 3)을 넣었을 때 3 과 가까운 local optima 가 나왔다. secant 함수에 newton 에서 수렴에 실패한 초기값이 포함되는 구간을 대입했을 때 optima 를 찾는 것에 실패했다. 반면 newton 에서 수렴에 성공한 초기값이 포함된 구간을 대입했을 때 global 또는 local optima 를 찾는데 성공했다. 특히 secant 에서도 newton 에서 global optima 의 수렴에 성공한 초기값 0 과 4.7 이 포함된 구간인 (-2, 1)과 (4.5, 5)이 초기값일 때 global optima 의 수렴에 성공하였다.

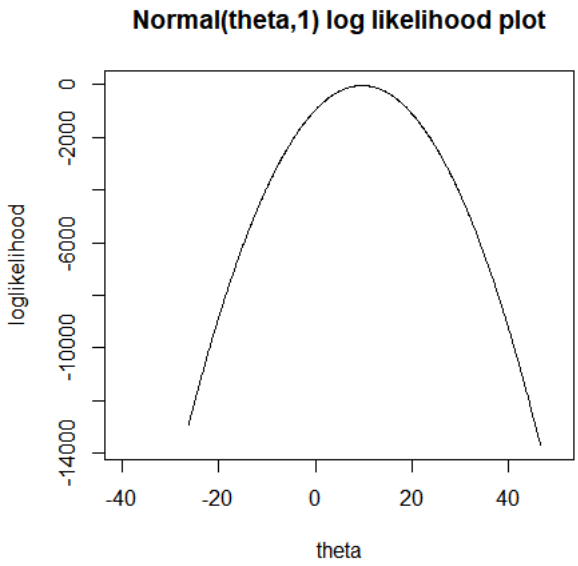
(e)

위의 (a)~(d)의 예제를 통해 수렴의 안정성은 bisection, newton, 그리고 secant 순이다. 이때 수렴의 안정성은 초기값에 무관하게 optima 를 찾는지에 대한 여부로 판단할 수 있다. 수렴의 속도는 newton, secant 그리고 bisection 순이다.

2.1-(e)

$$f(x;\theta) \sim \text{Norm}(\theta, 1)$$

$$\ell(\theta; x) = logL(\theta; x) = \sum_{i=1}^{n} logf(x_i; \theta, 1)$$



newton

initial value	-11	-1	0	1.5	4	4.7	7	8	38
θ^*	9.642	9.642	9.642	9.642	9.642	9.642	9.642	9.642	9.642
$\ell(\theta^*)$	-28.879	-28.879	-28.879	-28.879	-28.879	-28.879	-28.879	-28.879	-28.879
iteration	3	3	3	3	3	3	3	3	3
time	0.0015	0.0025	0.0033	0.0021	0.0019	0.0018	0.0031	0.0027	0.0019

newton 방법을 사용하였을 때 초기값에 무관하게 global optima 로 수렴한다. 초기값에 따라 수렴의 속도는 조금씩 차이가 있다.

bisection

initial value	[-1, 1]	[-1, 5]	[-20, -10]
θ^*	0.999	9.642	-10.000
$\ell(\theta^*)$	-775.704	-28.879	-3886.923
iteration	34	37	33
time	0.0223	0.0213	0.0182

bisection 은 초기값의 구간 안에서 반드시 하나의 local optima 를 찾는 알고리즘으로 안정적으로 수렴한다는 장점이 있지만 구간 안에 optima 가 없을 경우 optima 와 무관한 값을 산출한다. 따라서 bisection 을 통해 optima 를 구하기 위해서는 초기값이 반드시 optima 가 존재할 것으로 예상되는 구간이어야 한다.

secant

initial value	(-2, -1)	(-3, 3)	(-11, -10)	(3, 5)	(4.5, 5)	(6.5, 7.1)	(7, 9)	(37, 39)
θ^*	9.642	9.642	9.642	9.642	9.642	9.642	9.642	9.642
$\ell(\theta^*)$	-28.879	-28.879	-28.879	-28.879	-28.879	-28.879	-28.879	-28.879
iteration	3	2	3	3	3	2	2	3
time	0.0041	0.0021	0.0035	0.0029	0.0031	0.0017	0.0017	0.0028

secant 방법을 사용하였을 때 초기값에 무관하게 global optima 로 수렴한다. 초기값에따라 수렴의 속도가 조금씩 차이가 있다. 주목해볼 만한 점은 초기값에 따라 secant 가 newton 보다 반복횟수가 적거나 수렴 속도가 빠르다는 점이다.

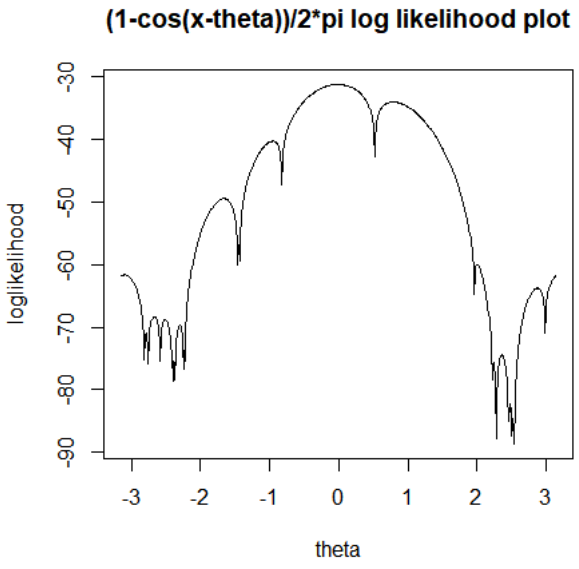
Norm$(\theta, 1)$은 log concave 하며 단봉 함수이다. Newton 과 Secant 을 사용하였을 때 초기값에 무관하게 global optima 로 수렴하였다. 반면에 bisection 은 초기값에 따라 최종 수렴된 optima 가 달랐는데 그 이유는 bisection 방법은 초기값으로 주어진 구간 안에서 optima 를 찾기 때문이다. 따라서 bisection 의 경우 global optima 가 존재할 것으로 예상되는 구간을 초기값일 때 안정적으로 global optima 에 수렴한다. 또한 일반적으로 수렴의 속도는 newton, secant, 그리고 bisection 순이지만 Norm$(\theta, 1)$과 같은 global optima 가 뚜렷이 구분되는 함수의 경우 secant 와 newton 의 수렴 속도 및 반복 횟수가 비슷하거나 초기값에 따라서는 secant 가 newton 보다 우위를 보이기도 하였다.

ex 2.2

2.2-(a)

$$f(x;\theta) = \frac{1 - cos(x-\theta)}{2\pi}$$

$$\ell(\theta; x) = \log L(\theta; x)$$
$$= \sum_{i=1}^{n} logf(x_i;\theta)$$
$$= \sum_{i=1}^{n} log\{1 - cos(x_i - \theta)$$



(b) Method of moment estimator

Method of moment estimator $\tilde{\theta}$는 평균의 함수가 표본의 평균과 같다고 식을 세운 후 θ에 대해 풀어서 구할 수 있다. 첫번째로,

$$\mu(\theta) = \frac{1}{2\pi}\int_0^{2\pi} x[1 - cos(x-\theta)]dx = \pi - \frac{1}{2\pi}\int_0^{2\pi} xcos(x-\theta)dx$$

부분 적분 후 $\sin(-x) = \sin(x)$과 $\sin(2\pi + x) = \sin(x)$의 성질을 이용하여 식을 정리하면 다음과 같은 식을 얻을 수 있다.

$$\int_0^{2\pi} xcos(x-\theta)dx$$

$$= xsin(x-\theta)|_0^{2\pi} - \int_0^{2\pi} sin(x-\theta)dx = 2\pi sin(2\pi - \theta)$$

$$\therefore \mu(\theta) = \pi - \frac{1}{2\pi}\int_0^{2\pi} xcos(x-\theta)dx = \pi - sin(\theta)$$

$\mu(\theta) = \bar{X}$라고 놓고 θ에 대해 풀면

$$\mu(\theta) = \bar{X} \iff \pi - sin(\theta) = \bar{X} \iff \arcsin(\pi - \bar{X})$$

따라서 Method of Moments estimator 인 $\tilde{\theta}$는 $\arcsin(\pi - \bar{X}) = -0.05844061$이 된다.

(c) Newton method

initial value	MOM=-0.05844061	-2.7	2.7
θ^*	-0.012	-2.667	2.873
$\ell(\theta^*)$	-31.343	-68.399	-63.806
iteration	10	5	6
time	0.0041	0.0028	0.0035

(d) partition interval using newton

$\hat{\theta}_i$	Lower	Upper	$\hat{\theta}_i$	Lower	Upper
-3.093	-3.142	-2.826	-0.953	-1.405	-0.837
-2.78616675160335	-2.794	-2.794	-0.012	-0.805	0.489
-2.78616675160402	-2.763	-2.763	**0.791**	**0.521**	**1.942**
-2.667	-2.731	-2.605	**2.004**	**1.973**	**2.194**
-2.508	-2.573	-2.415	2.236	2.226	2.258
-2.388	-2.384	-2.384	2.361	2.289	2.447
-2.297	-2.352	-2.258	2.475	2.479	2.479
-2.232	-2.226	-2.226	2.514	2.510	2.510
-1.658	-2.194	-1.468	2.873	2.542	2.984
-1.447	-1.437	-1.437	3.190	3.015	3.141

$[-\pi, \pi]$ 를 200 개의 구간으로 나누어 newton 을 이용해 optimization 한 결과 20 개의 unique 한 optima 로 수렴하는 초기값 구간을 나눌 수 있다. 특히 [ ] 표시된 두 구간의 경우 초기값이 인접함에도 불구하고 수렴된 optima 의 값은 0.791 에서 2.004 로 크게 점프한다.

(e) Find two starting values, as nearly as you can, which result in different optima by using newton

initial value	1.95997366523361	1.96002637607754
θ^*	0.791	2.004
$\ell(\theta^*)$	-34.141	-59.997
iteration	19	17
time	0.0075	0.0096

(d)번의 [ ] 구간은 초기값이 인접하지만 수렴된 optima 의 값이 크게 점프한다. 최대한으로 optimization 이 가능한 범위까지 두 구간의 거리를 좁힌 결과 초기값이 1.95997366523361 일 때는 0.791 에 수렴한 반면 초기값이 1.96002637607754 일 때는 2.004 에 수렴하였다. 두 초기값의 차이는 0.0005271084393010 38 이다. 이처럼 매우 근접한 값임에도 불구하고 newton 의 optimization 알고리즘에 의해 서로 다른 optima 로 수렴한다는 것을 알 수 있다.

R 에서 Step 함수의 scope argument 에 대하여

Subset selection 은 통계 모델링을 할 때 반응 변수를 잘 예측할 수 있는 설명 변수들의 조합을 찾는 과정이다. 본 과제에서는 Subset selection 의 다양한 방법론 중에서 주로 lm 과 glm 에서 사용되는 R 의 step 함수에 대해 다룰 것이다. Step 함수는 stepwise 알고리즘에 따라 AIC 값이 가장 낮은 모델의 설명 변수 조합을 구한다. Stepwise 알고리즘은 모델에서 설명 변수들을 반복적으로 추가하거나

제거하는 방법으로 가장 모델의 성능을 높여주는 설명 변수들의 조합을 구한다. 이때 모델의 성능은 AIC 값이 낮을수록 높아진다.

Stepwise 알고리즘은 크게 forward selection, backward elimination, 그리고 stepwise 세가지 방법으로 나뉜다. Forward Selection 은 설명 변수가 없고 오직 intercept 항만 존재하는 모델에서 시작해서 유의한 설명 변수들을 차례로 하나씩 추가한다. Forward Selection 의 단점은 이미 모델에 존재하는 설명 변수가 새롭게 추가된 설명 변수에 의해 유의성이 사라지더라도 모델에서 제거하지 못하는 것이다. Backward Elimination 은 설명 변수들 전체가 적합된 모델에서 시작해 유의하지 않은 설명 변수들을 차례로 제거한다. 그러나 한 번 제거된 설명 변수는 다시 모델에 추가될 수 없다는 단점이 있다. 마지막으로 stepwise 는 forward selection 과 backward elimination 을 결합한 방법으로 R 에서는 step 함수의 object argument 의 모델에서 시작해 유의한 변수를 반복적으로 추가하거나 제거해 가장 이상적인 설명 변수들의 조합을 구한다.

R 의 step 함수는 local search 방법론 중 하나로 제한된 시간 안에 경쟁력 있는 local optima 를 찾는다. Step 함수의 scope 라는 argument 는 stepwise 에서 고려될 모델의 local search 범위를 지정해준다. Argument 에는 local search 의 최대 범위인 upper 와 최소 범위인 lower 로 지정된 list 형식이 들어간다. 최종 수렴 모델은 lower 를 반드시 포함하는 것과 동시에 upper 에 포함된다. Scope 에 single formula 만 주어진 경우에는 lower 는 생략되고 single formula 가 upper 지정된다. Scope argument 가 생략될 경우에는 lower 는 생략되고 step 함수의 object argument 에 주어진 모델이 upper 가 된다. Scope 에서 모델을 지정해 줄 때 알아 두어야 할 약어가 있다. ~.는 object 에서 이미 지정된 모델을 의미하고 ~.^2 는 object 에서 지정된 모델의 항들과 각 항들의 교호작용 항들을 의미한다.

R 의 base 데이터 중 하나인 mtcars 를 이용하여 scope 를 적용해보자. mtcars 는 mpg 라는 하나의 반응변수와 cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb 라는 10 개의 설명변수로 이루어진 데이터이다. 사용자가 지정한 모델에서부터 local search 를 시작하고 싶다면 object argument 에 출발 모델을 지정하고 scope 에 local search 범위를 lower 과 upper 에 지정해준다. 이때 null 은 intercept 만 포함된 모델이고 full 은 모든 설명 변수들의 단일항이 포함된 모델이며 fit1 은 사용자가 지정한 모델로 mpg ~ cyl + disp + hp 이다.

```
step(fit1, scope=list(lower=null, upper=full), direction="both")
```

```
## Start:  AIC=75.21
## mpg ~ cyl + disp + hp


## Step:  AIC=63.53
## mpg ~ cyl + disp + hp + wt
```

```
##
## Step:  AIC=62.66
## mpg ~ cyl + hp + wt


## Call:
## lm(formula = mpg ~ cyl + hp + wt, data = mtcars)
##
## Coefficients:
## (Intercept)          cyl           hp           wt
##    38.75179     -0.94162     -0.01804     -3.16697
```

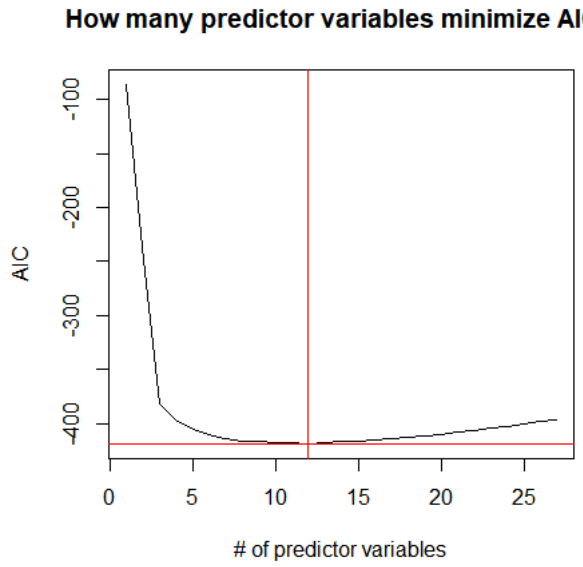
사용자가 object argument 에 지정해준 fit1(mpg ~ cyl + disp + hp) 모델에서 출발하여 full 에 존재하는 모든 설명 변수들을 모델에 추가 및 제거한 결과 최종적으로 수렴한 모델을 제시한다. 만약 사용자가 지정한 모델의 설명 변수들을 모두 포함하는 최종 수렴 모델을 구하기 위해서는 scope 의 lower 에 사용자 지정 모델을 입력한다.

```
step(fit1, scope=list(lower=fit1, upper=full), direction="both")

## Start:  AIC=75.21
## mpg ~ cyl + disp + hp
##
## Step:  AIC=63.53
## mpg ~ cyl + disp + hp + wt
##
## Call:
## lm(formula = mpg ~ cyl + disp + hp + wt, data = mtcars)
##
## Coefficients:
## (Intercept)          cyl         disp           hp           wt
##    40.82854     -1.29332      0.01160     -0.02054     -3.85390
```

위와 같이 lower 에 사용자 지정 모델인 fit1(mpg ~ cyl + disp + hp)를 입력하면 최종 수렴 모델에 fit1 의 설명 변수들이 포함된다. 따라서 위와 같은 R output 을 통해 scope argument 의 local search 범위 안에서 stepwise 방법으로 model 을 build up 하는 원리를 이해할 수 있다.

Baseball players' salary regression model selection problem



주 효과만 고려한 선형 모델 중 best subset 을 찾기 위해 총 4 가지의 방법–All possible subsets, stepwise, backward elimination, 그리고 forward selection– 을 사용하였다. 이때 best subsets 의 기준은 AIC 가 가장 작은 모델이다. All possible subsets 는 R 의 leaps 패키지에 있는 regsubsets 함수를 사용하였다. regsubsets 의 아웃풋은 AIC 값이 없기 때문에 RSS 를 이용해 다음과 같이 계산하였다.

$$AIC = N * \log(RSS/N) + 2(p + 1)$$

where N is sample size and p is number the number of predictors

위의 그래프는 regsubsets 을 이용해 모델의 설명 변수 개수에 따른 AIC 값을 그림으로 나타낸 것이다. 그 결과 설명 변수가 12 개일 때 AIC 값이 최소로 나온다는 것을 확인할 수 있다. 나머지 stepwise, backward 그리고 forward 방법은 R 의 step 함수를 이용하였으며 수렴 모델의 AIC 값은 아래 표를 통해 확인할 수 있다.

	Predictors Selected																		
Method	1	2	3	4	6	8	9	10	13	14	15	16	20	21	22	24	25	26	AIC
All		●	●		●	●		●	●	●	●	●				●	●	●	-418.9472
Stepwise	●		●		●	●		●	●	●	●	●				●	●	●	-418.9421
Backward	●		●		●	●		●	●	●	●	●				●	●	●	-418.9421
Forward			●	●		●	●	●	●	●	●		●	●	●	●	●		-413.0164

위 표에 따르면 4 가지 방법을 통해 최종 optimal 모델을 구한 결과 all possible subset 을 통해 가장 minimum 한 AIC 값을 가진 subset 을 구할 수 있으며 각 모델에 어떤 설명 변수들이 선택되었는지 알 수 있다. 설명 변수의 수는 Forward 가 13 개로 가장 많고 all, stepwise, 그리고 backward 는 모두

12 개이다. Stepwise 방법과 backward 방법은 최종 수렴 모델이 같다. 따라서 Stepwise 와 backward 의 AIC 값이 같은데 반면에 All possible subsets 의 최종 수렴 모델은 backward 와 stepwise 의 최종 수렴 모델과 달리 설명변수 1 번대신 2 번을 선택함으로써 AIC 값을 최소화해 더 optimal 한 해답을 구했다. 따라서 최종 모형은 AIC 의 값이 가장 낮은 all possible subset 방법으로 local search 한 모델이며 결과는 다음과 같다.

$$\begin{aligned}\log(\text{salary}) = {} & 5.293 - 1.345 * \text{obp} + 0.01603 * \text{runs} - 0.02625 \\ & * \text{triples} + 0.01048 * \text{rbis} - 0.00441 * \text{sos} + 1.508 \\ & * \text{freeagent} + 1.327 * \text{arbitration} - 0.3691 * \text{runsperso} \\ & + 0.1502 * \text{histperso} - 0.00015 * \text{soserrors} + 0.07731 \\ & * \text{sbsobp} - 0.00028 * \text{sbsruns}\end{aligned}$$

설명 변수들의 다중공선성을 고려하지 않았기 때문에 계수의 절대치에 대한 해석 보다는 계수의 음수와 양수에 따라 음과 양의 상관관계의 해석에 중심을 둘 것이다. 위의 모델에 따르면 야구선수의 득점(runs), 타점(rbis), 자유 계약 선수(freeagent), 연봉 조정(arbitration), histperso(안타/실책) 그리고 sbbsobp(도루*출루율)이 로그 연봉과 양의 상관관계를 띄며 그 중 자유 계약 선수(freeagent)의 계수 절대치가 가장 크다. 그 다음으로 절대치가 큰 계수는 연봉 조정(arbitration)이다. 이는 프로야구 시즌이 끝난 후 야구 선수들은 구단을 떠나 자유 계약 선수가 되어 다른 구단으로 이동하면서 연봉 협상을 통해 더 높은 연봉을 받기 때문이다. 반면에 야구선수의 출루율(obp), 3 루타(triples), 삼진(sos), 득점/삼진(runsperso), 삼진*실책(soserrors), 그리고 도루*득점(sbsruns)는 로그 연봉과 음의 상관관계를 띄며 그 중 득점/삼진(rusperso)의 계수 절대치가 가장 크다.

3. Discussion

이번 과제를 통해 bisection, newton 그리고 secant 알고리즘을 optima 가 많은 다봉 함수에서 실현할 수 있었다. 예상했던 것과 같이 세 알고리즘 모두 초기값에 따라 다양한 optima 수렴 양상을 보였다. 일반적으로 수렴 속도는 newton, secant, 그리고 bisection 순이었고 수렴의 안정성은 bisection, newton 그리고 secant 순이었다. secant 의 초기값이 newton 에서 수렴에 실패한 초기값이 포함된 구간일 때 secant 는 수렴에 실패하였다. 또한 newton 을 이용해 global optima 를 구하기 위해 초기값을 데이터를 대표하는 평균 값으로 지정했을 때 오히려 수렴에 실패하였다. 대개 초기값이 global optima 에 인접하면 newton 방법을 통해 global optima 에 수렴하는 편이었지만 몇 가지 경우에는 초기값이 global optima 에 인접함에도 불구하고 수렴에 실패하거나 다른 local optima 로 수렴하였다. normal 분포처럼 log concave 하고 smooth 한 단봉 함수의 경우 newton 과 secant 는 초기값에 무관하게 global optima 에 수렴하였다. 반면에 bisection 은 newton 과 secant 와 달리 업데이트 값이 초기값의 구간 바깥으로

이동하지 않기 때문에 초기값 구간안에 global optima 가 없을 경우 global optima 로 수렴에 실패한다. 일반적으로 newton 이 secant 보다 반복 횟수가 적고 수렴 속도가 빠르다고 알려진 것과 달리 normal 분포의 최적화에서는 secant 가 newton 보다 수렴 속도와 횟수에서 우위를 보이기도 했다.

경우의 수 최적화 문제 중 하나인 regression model selection 또는 best subsets 을 구하는 문제를 이번 과제에서 다루었다. 모든 경우의 수를 고려하는 대신 제한된 시간안에 경쟁력 있는 local optima 를 구하기 위해 다양한 local search 를 이용할 수 있다. 그 중에 all possible subset, stepwise, backward elimination 그리고 forward selection 을 사용해 optimal 모델을 구했다. 그 결과 Baseball Players' Salaries 데이터의 경우 all possible subsets 방법을 이용했을 때 가장 AIC 값이 낮은 모델을 구할 수 있었다. AIC 값 뿐만 아니라 설명 변수들의 다중공선성과 같은 이슈에 대해 추가적인 논의가 수반될 경우 더 나은 모형으로 개선시킬 수 있을 것이다.

4. Appendix

```
rm(list=ls())
library(numDeriv)
library(tidyverse)
options(scipen=999)

### Bisection function
bisec<-function(f, a, b){

  maxiter <- 1000
  threshold <- 10^(-10)
  err <- 1
  niter <- 0
  x0 <- ( a + b ) / 2

  while ( niter <= maxiter && err >= threshold){
    #update interval
    if ( (genD(func = f, x = a)$D[1]) * (genD(func = f , x = x0)$D[1]) <=0)     {b <- x0}
    else {a <- x0}

    #update x
    oldx0 <- x0
    x0 <- ( a + b )/2


    #update error and niter
    err<-abs( oldx0 - x0 )
    niter <- niter + 1

  }
  print(paste(paste0("Number of itertation is :",niter),paste0("x* is :",x0),paste0("f(x*)
is : ",f(x0)),sep="      "))
}

### Newton's function
newt <- function( f, x0 ){

  maxiter <- 1000
```

```
  threshold <- 10^(-10)
  err <- 1
  niter <- 0

  while ( niter <= maxiter && err >= threshold){

    #update x
    oldx0 <- x0
    x0 <- oldx0 - ( genD(f, oldx0)$D[1] / genD(f, oldx0)$D[2] )


    #update error and niter
    err<-abs( oldx0 - x0 )
    niter <- niter + 1

  }
  print(paste(paste0("Number of itertation is :",niter),paste0("x* is :",x0),paste0("f(x*)
is : ",f(x0)),sep="      "))
}

### secant function
sec <- function( f, x0, x1 ){

  maxiter <- 1000
  threshold <- 10^(-10)
  err <- 1
  niter <- 0
  x2 <- 0

  while ( niter <= maxiter && err >= threshold){

    # calculate x2
    x2 <- x1 - genD(f, x1)$D[1] / ((genD(f, x1)$D[1] - genD(f, x0)$D[1])/(x1 - x0))

    # update x0 and x1
    x0 <- x1
    x1 <- x2


    #update error and niter
    err <- abs( x1 - x0 )
    niter <- niter + 1

  }
  print(paste(paste0("Number of itertation is :",niter),paste0("x* is :",x1),paste0("f(x*)
is : ",f(x1)),sep="      "))
}


### HW2-1. 교재 예제 풀기
# ex 2.1 (a)
xx <- c(1.77, -0.23, 2.76, 3.80, 3.47, 56.75, -1.34, 4.24, -2.44, 3.29, 3.71, -2.40, 4.53, -
0.07, -1.05,
        -13.87, -2.53, -1.75, 0.27, 43.21)
cauchy.ll <- function(x){
  sum(log(dcauchy(x=xx,location=x,scale=1)))
}
tht <- seq(-100,100,by=0.1)
plot(tht, sapply(X=tht, FUN=function(x)cauchy.ll(x)),type='l',xlab="theta",
ylab="loglikelihood", main="Cauchy(theta,1) log likelihood plot") # graph of loglikelihood
```

```
system.time(for(i in 1:100)newt(f=cauchy.ll, x0=-11))
system.time(for(i in 1:100)newt(f=cauchy.ll, x0=-1))
system.time(for(i in 1:100)newt(f=cauchy.ll, x0=0))
system.time(for(i in 1:100)newt(f=cauchy.ll, x0=1.5))
system.time(for(i in 1:100)newt(f=cauchy.ll, x0=4))
system.time(for(i in 1:100)newt(f=cauchy.ll, x0=4.7))
system.time(for(i in 1:100)newt(f=cauchy.ll, x0=7))
system.time(for(i in 1:100)newt(f=cauchy.ll, x0=8))
system.time(for(i in 1:100)newt(f=cauchy.ll, x0=38))
system.time(for(i in 1:100)newt(f=cauchy.ll, x0=mean(xx)))

# ex 2.1 (b)
system.time(for(i in 1:100)bisec(f=cauchy.ll, -1, 1))
system.time(for(i in 1:100)bisec(f=cauchy.ll, -1, 5))
system.time(for(i in 1:100)bisec(f=cauchy.ll, -101, -102))

# ex 2.1 (d)
system.time(for(i in 1:100)sec(f=cauchy.ll, -2, -1))
system.time(for(i in 1:100)sec(f=cauchy.ll, -3, 3))
system.time(for(i in 1:100)sec(f=cauchy.ll, -11, -10))
system.time(for(i in 1:100)sec(f=cauchy.ll, 3, 5))
system.time(for(i in 1:100)sec(f=cauchy.ll, 4.5, 5))
system.time(for(i in 1:100)sec(f=cauchy.ll, 6.5, 7.1))
system.time(for(i in 1:100)sec(f=cauchy.ll, 7, 9))
system.time(for(i in 1:100)sec(f=cauchy.ll, 37, 39))

# ex. 2.1 (e)
teta <- seq(-40,50,by=0.1)
samp <- rnorm(20, mean=10, sd=1)
norm.ll <- function(x){
  sum(log(dnorm(x=samp,mean=x,sd=1)))
}
plot(teta, sapply(X=teta, FUN=function(x)norm.ll(x)),type='l',xlab="theta",
ylab="loglikelihood", main="Normal(theta,1) log likelihood plot") # graph of loglikelihood

# newton method
system.time(for(i in 1:100)newt(f=norm.ll, x0=-11))
system.time(for(i in 1:100)newt(f=norm.ll, x0=-1))
system.time(for(i in 1:100)newt(f=norm.ll, x0=0))
system.time(for(i in 1:100)newt(f=norm.ll, x0=1.5))
system.time(for(i in 1:100)newt(f=norm.ll, x0=4))
system.time(for(i in 1:100)newt(f=norm.ll, x0=4.7))
system.time(for(i in 1:100)newt(f=norm.ll, x0=7))
system.time(for(i in 1:100)newt(f=norm.ll, x0=8))
system.time(for(i in 1:100)newt(f=norm.ll, x0=38))

# bisection
system.time(for(i in 1:100)bisec(f=norm.ll, -1, 1))
system.time(for(i in 1:100)bisec(f=norm.ll, -10, 10))
system.time(for(i in 1:100)bisec(f=norm.ll, -20, -10))

# secant
system.time(for(i in 1:100)sec(f=norm.ll, -2, -1))
system.time(for(i in 1:100)sec(f=norm.ll, -3, 3))
system.time(for(i in 1:100)sec(f=norm.ll, -11, -10))
system.time(for(i in 1:100)sec(f=norm.ll, 3, 5))
system.time(for(i in 1:100)sec(f=norm.ll, 4.5, 5))
system.time(for(i in 1:100)sec(f=norm.ll, 6.5, 7.1))
system.time(for(i in 1:100)sec(f=norm.ll, 7, 9))
system.time(for(i in 1:100)sec(f=norm.ll, 37, 39))
```

```
# ex. 2.2 (a)
tth <- seq(-pi,pi,by=0.01)
smp <- c(3.91, 4.85, 2.28, 4.06, 3.70, 4.04, 5.46, 3.53, 2.28, 1.96, 2.53, 3.88, 2.22, 3.47,
4.82, 2.46, 2.99, 2.54, 0.52, 2.5)
llh <- function(x){
  sum(log((1-cos(smp-x))/(2*pi)))
}
plot(tth, sapply(X=tth, FUN=function(x)llh(x)),type='l',xlab="theta", ylab="loglikelihood",
main="(1-cos(x-theta))/2*pi log likelihood plot") # graph of loglikelihood

# ex. 2.2 (b)
mom <- asin(pi-mean(smp))

# ex. 2.2 (c)
system.time(for(i in 1:100)newt(f=llh, x0=mom))
system.time(for(i in 1:100)newt(f=llh, x0=-2.7))
system.time(for(i in 1:100)newt(f=llh, x0=2.7))

# ex. 2.2 (d)
spc <- seq(-pi,pi,length.out = 200)
interval <- list()
for(i in 1:length(spc))interval[[i]] <- newt(f=llh, x0=spc[i])
names(interval) <- spc
interval

# ex. 2.2 (e)
spc2 <- seq(1.94178842407811,1.97336221959158,length.out = 600)
interval2 <- list()
for(i in 1:length(spc2))interval2[[i]] <- newt(f=llh, x0=spc2[i])
names(interval2) <- spc2
interval2

system.time(for(i in 1:100)newt(f=llh, x0=1.95997366523361))
system.time(for(i in 1:100)newt(f=llh, x0=1.96002637607754))

# step 함수의 scope 에 대하여
names(mtcars) # mpg 가 하나의 종속변수 그리고 나머지 10 개의 설명변수

# search range
full <- lm(mpg~., mtcars)
null <- lm(mpg~1, mtcars)
fit1 <- lm(mpg~cyl+disp+hp, mtcars)

# scope 가 upper lower 모두 지정 되어있고 모델을 중간에서부터 build up 하고 싶을 때
step(fit1, scope=list(lower=null, upper=full), direction="both")
step(fit1, scope=list(lower=fit1, upper=full), direction="both")

# scope 가 single formula 인 경우
step(fit1, scope=mpg~cyl+disp+hp+wt+qsec+vs, direction="both")

# scope 가 없는 경우
step(fit1, direction = "both")

# ~. 사용
step(fit1, scope=~., direction = "both")
```

```
# ~.^2 사용
step(fit1, scope=~.^2, direction = "both")


# Baseball players' salary model local search
library(MASS); library(leaps)

ball <- read.table('C:/Users/dnskd/Desktop/19-2/계특/과제/hw2/baseball.txt',header=T)

# log of salary as response variable
ball$ln_salary <- log(ball$salary)
ball <- ball[,-1]

# all possible subset
reg <- regsubsets(ln_salary ~., data=ball, nvmax = 27)                          # all possible
sum.reg <- summary(reg)
p <- c(1:27)
aic <- nrow(ball)*log(sum.reg$rss/nrow(ball))+2*(p+1)
plot(p, aic, xlab="# of predictor variables", ylab="AIC", main="How many predictor variables
minimize AIC?", type='l')
abline(h=min(aic),v=which.min(aic), col='red')
allpop <- lm(ln_salary ~
obp+runs+triples+rbis+sos+freeagent+arbitration+runsperso+hitsperso+soserrors+sbsobp+sbsruns,d
ata=ball)
extractAIC(allpop)                                                              # -418.9472

# stepwise / backward elimination/ forward selection
lm_full <- lm(ln_salary ~ . , data=ball)
lm_null <- lm(ln_salary ~ 1 , data=ball)

step(lm_full, direction="both",trace=0)
lm_both <- lm(formula = ln_salary ~ average + runs + triples + rbis + sos +
                freeagent + arbitration + runsperso + hitsperso + soserrors +
                sbsobp + sbsruns, data = ball)
extractAIC(lm_both)                                                             # -418.9421

step(lm_full, direction = "backward", trace=0)
lm_back <- lm(formula = ln_salary ~ average + runs + triples + rbis + sos +
                freeagent + arbitration + runsperso + hitsperso + soserrors +
                sbsobp + sbsruns, data = ball)
extractAIC(lm_back)                                                             # -418.9421

step(lm_null, direction = "forward", scope=list(lower=lm_null, upper=lm_full), trace=0)
lm_for <- lm(formula = ln_salary ~ hits + freeagent + arbitration + rbis +
               soserrors + sbsobp + sos + walks + runsperso + obppererror +
               hitspererror + runs + runspererror + sbsruns, data = ball)
extractAIC(lm_for)                                                              # -413.0164
```