Homework #5

Implement Conjugate gradient Methods & Disscusion

GIST AI

20231126 황산하

*코드는 Python으로 작성했고 Report와 함께 Zip으로 압축하여 제출하였습니다.

INDEX

1. Comparison and Result

본 과제는 Conjugate Gradient (CG)의 최적화 방법을 구현하고 Non Linear 방법인 FR, PR 그리고 HS 의 성능을 비교하는 과제이다. Linear를 가정하고 Local Minimum을 찾는 Linear CG 방법과 Non Linear를 가정하고 Local Minimum을 찾는 Non Linear CG 각각의 특징 때문에 함수의 특징을 고려해서 알고리즘을 사용해야 하는데, 그래서 Linear CG 방법에서 실험한 함수와 Non Linear에서 사용한 함수가 달라서 직접적인 비교는 어렵다. 따라서 본 과제에서는 Non linear CG 방법들 사이에서의 비교를 하고 Linear CG는 결과만 보고서에 기록하려고 한다. Non linear CG는 각각 β를 구하는 방식에서 차이가 있는데, 생각보다 안정성 면에서, 속도면에서 차이가 존재했다. Non linear CG를 비교할 땐 명확한 비교를 위해 초기점을 (0.5,0.5)로 통일했고, 동일한 종결조건을 사용했다. Step size를 구할 땐 이전과제에서 구현한 inexact search와 wolfe condition을 적용한 inexact search를 이용했다. 사용한 함수는 이전 과제의 함수와 같다.

$$F_1(x,y) = (x+3y-5)^2 + (3x+y-7)^2$$

$$F_2(x,\ y) = 50(x-y^2)^2 + (1-y)^2$$

$$F_3(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$$

1) Linear Conjugate Gradient Method (Result Only)

[표 1 Linear Conjugate 결과표]

Linear	(x, y)	Iter	Convergence	f(x,y)
F1	2.01631, 0.965561	99	Yes	0.00777987

초기 점은 (0,0)으로 잡고 알고리즘을 구현했을 때 얻을 수 있는 결과이다. 이전 과제 4에서 했던 gradient based 알고리즘들과 마찬가지로 step size가 성능에 영향을 끼치는 것 같았다. Step size를 search 하는 알고리즘으로는 wolfe를 적용한 line search 와 inexact line search를 비교했다. 둘다 잘 수렴하는 모습을 보였다.

2) Non-Linear Conjugate Gradient Method : FR

[표 2 FR결과표]

FR	(x, y)	Iter (Time)	Convergence	f(x,y)
F2	0.352, 0.597	4 iter (0.003sec)	Yes	0.163
F3	2.50, 0.319	6 iter (0.0049sec)	Yes	0.08

FR 방법은 현재의 gradient와 이전의 gradient의 비율로 β를 구하는 방식을 사용한다. 어느 gradient method들과 마찬가지로 초기치에 민감한 모습을 보였으며, 잘 수렴하다가 갑자기 발산하며 종료가 되길래 너무 크게 발산하면 이전의 결과값을 local minimum으로 사용하는 종결조건을 이용했다. 다른 Non linear CG 방법들에 비해 빠르게 수렴하는 특징을 보였다. 수렴된 결과값들을 확인해 보면 다른 방법들에 비해 더 좋은 local minimum으로 수렴했다는 것을 확인할 수 있다.

3) Non-Linear Conjugate Gradient Method : PR

[표 3 PR 결과표]

PR	(x, y)	Iter (Time)	Convergence	f(x,y)
F2	0.281, 0.633	2 iter (0.00199sec)	Yes	0.855

F3	0.744,-1.4533	611iter (0.00399sec)	Yes	9.74

PR 방법 역시 초기값과 Step size에 민감한 모습을 보였다. 초기값을 정말 잘 준 경우에는 한번의 iteration 만에 종결조건을 만족해서 알고리즘이 종료될 때도 있었으며 결과표를 보면 알 수 있듯이 611 번의 반복 이후에 local minimum을 찾을 때도 있었다. 611번의 반복을 하는 만큼 FR 방법에 비해서 발산하지 않고 안정적으로 local minimum에 수렴하는 특징을 가지고 있지만 경우에 따라서는 PR 방법보다 느릴 수 있다는 특징을 가진다. 혹여나 발산할까 FR 방법에서 사용했던 너무 크게 발산하면 종료하고 이전 결과값을 받아오는 종료조건을 사용했으나 그 종료조건에 의해서 종료 되지는 않았다.(안정적)

FR방법과 비슷하게 β를 구할 때 이전 gradient와 갱신된 gradient의 비율로 사용하지만, 이전 gradient와 현재 gradient의 차이를 구하는 항을 삽입하여 비율을 구함으로써 차이가 크면 β가 커지고, 차이가 작으면 β가 작아지는 매우 직관적인 방법을 이용했다. 이런 단순한 아이디어가 안정성을 추가했다는 사실이 흥미로운 점이다. 하지만 결과값을 다른 방법들과 비교해보면 조금 보수적으로 이동하는 만큼 더 좋은 local minimum으로 수렴했다고 보기는 어려워 보인다.

4) Non-Linear Conjugate Gradient Method : HS

[표 4 HS 결과표]

HS	(x, y)	Iter (Time)	Convergence	f(x,y)
F2	0.39147, 0.71453	6 iter (0.004sec)	Yes	0.79
F3	1.6585, -0.6311	40 iter (0.0451sec)	Yes	3.32

HS 방법은 분모에도 이전 gradient와 현재 gradient의 차이를 구하는 항을 삽입하는 식으로 β를 구한다. 처음에 생각했을 땐 FR방법이랑 이 방법이 큰 차이가 없을 거라고 생각했는데, 생각보다 결과는 달랐다. FR 방법보다는 느리지만 안정적으로 수렴했고, PR보다는 빠르게 local minimum으로 수렴했다. 결과값을 확인해봐도 비슷한 결과를 보이는데 FR보다는 좋지않고 PR보다는 더 좋은 local minimum으로 수렴했다. 따라서 더 좋은 local minimum을 찾는 FR의 장점과 안정적으로 수렴하는 PR의 장점을 합쳐놓은 방법이라고 볼 수 있다.

2. Disccusion - Terminal Condition

종결 조건으로는 세가지 방법들 모두 크게 발산하면 종료, 최대 반복횟수를 만족하면 종료, gradient의 norm이 0이되면 종료하는 종결 조건을 사용했다. 하지만 사용한 종결조건을 자세히 보면 norm을 구하는 방법에서 종료하기가 쉽지 않고(초기값이나 step size가 적절히 주어지지 않는다면 발산), 크게 발산하면 종료하는 방법은 개인적으로는 너무나 사람 직관적인 방법이라고 생각한다. 이전 과제들을 진행하면서 더 좋은 종결조건이 있다면 더 괜찮은 알고리즘을 만들 수 있을 거라는 생각을 하고 있지만 아직까지는 좋은 생각이 떠오르지 않는다. 아직까지는 이전값과 현재값을 고려해서 더이상 갱신이 안될 경우에 종료하는 조건이 가장 좋다고 생각하는데 알고리즘에서 발산하는 상황을 고려한다면 현재의 종료조건이 아직까지는 최선일 거라고 생각한다.

3. Appendix A : Code

코드는 Python으로 구현했으며 제출될 폴더 안에는 method.py, result.py, utils.py가 있다. 이전 과제와 마찬가지로 utils.py에는 알고리즘 구현을 위해 필요한 gradient를 구하는 함수와 inexact line search 함수가 구현되어있다. Method.py안에 각각의 CG알고리즘들이 구현되어 있다. Result.py를 실행 시킴으로써 결과를 받아 볼 수 있는데 스크립트에서 함수와 알고리즘을 선택해서 실행시켜야 한다. 예를 들어 python result.py 이런식으로 실행 시킬 수 있다. result.py 내부에서 main함수 내부의 mode를 linear_conjugate, cg_fr, cg_pr, cg_hs 중에서 선택해서 mode를 고를 수 있다. F 도 역시 선택해서 실행시키면 결과를 받아

볼 수 있다. (이전 과제에서 script 상에서 입력으로 고를 수 있다는 식으로 써놨는데 확인해보니 잘안되는거 같고 이번에 말한 방식으로 돌려야 돌아가는 것을 확인했습니다.)