Math 110A Project 1 Report

Implementing Steepest Gradient Descent

by Alicia Chuang, Tyler Brown, Vinachau Pham

February 2023

1 Steepest Gradient Descent

The Steepest Gradient Descent Method is a first order optimization algorithm for finding local minimizers. Since the gradient is the direction of most positive change, any differentiable function decreases the fastest in the direction of the negative gradient. Thus, for an initial guess $\mathbf{x}^{(0)}$, we can use the algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)})$$

with α_k being the step size of each iteration, to approximate the local minimizer $\mathbf{x}^*$. Convergence is reached once $\nabla f(\mathbf{x}^{(k)})$ is within the tolerated region of 0, since $f(\mathbf{x}^*) = 0$ by definition. The optimal α_k can be found by solving the one dimensional optimization problem

$$\alpha_k = \underset{\alpha>0}{\operatorname{argmin}}\, f(\mathbf{x}^{(k)} - \alpha f(\mathbf{x}^{(k)}))$$

2 Secant Method

Newton's method is one of the fastest converging optimization algorithms, but it requires the second derivative. In the case that the second derivative is not analytically computable, we may approximate $f''(x^{(k)})$ with

$$\frac{f'(\mathbf{x}^{(k)}) - f'(\mathbf{x}^{(k-1)}}{\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}}$$

Then, from the Newton's Method, we obtain the algorithm for the Secant Method:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \frac{\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}}{f'(\mathbf{x}^{(k)}) - f'(\mathbf{x}^{(k-1)})} f'(\mathbf{x}^{(k)})$$

3 Code and Output

Our code is written in python, with comments, on this github repository

Testing only the linesearch_secant function using $\alpha^2 - 20$ with initial guesses of $\alpha_0 = 2$ and $\alpha_1 = 8$, converging in 7 steps as expected:

```
Testing of the linesearch_secant function:
expected value: 4.4721359553
function result: 4.472135955300302
```

Testing the Rosenbrock's function $f(\mathbf{x}) = 100 * (\mathbf{x}_1 - \mathbf{x}_0^2)^2 + (1 - \mathbf{x}_0)^2$, with $\nabla f(\mathbf{x}) = \begin{bmatrix} 400 * \mathbf{x}_0^3 - 400 * \mathbf{x}_0 * \mathbf{x}_1 + 2 * \mathbf{x}_0 - 2 \\ 200 * (\mathbf{x}_1 - \mathbf{x}_0^2) \end{bmatrix}$, initial guess $\begin{bmatrix} 1.5 \\ 2 \end{bmatrix}$, tolerance 10^{-9}, initial guess $\alpha_0 = 0.01$ and $\alpha_1 = 0.011$, printing out every 20:

```
Rosenbrock's function:
[-0.65277988  0.42994197]
[0.26137121 0.07127204]
[0.49017724 0.24277243]
[0.62121999 0.38804387]
[0.71536728 0.51353978]
[0.78717717 0.6211185 ]
[0.84276297 0.71142626]
[0.88555209 0.78511768]
[0.91797321 0.84336661]
[0.94204621 0.8879605 ]
[0.95955392 0.92111056]
[0.97205023 0.94514102]
[0.98083149 0.96221124]
[0.98692152 0.97413838]
[0.99107657 0.98231761]
[0.99391173 0.98791851]
[0.99584854 0.99175392]
[0.99717112 0.99437728]
[0.99807349 0.99616913]
[0.99868863 0.99739155]
[0.99910767 0.99822469]
[0.99939296 0.99879211]
[0.99958711 0.99917836]
[0.9997192  0.99944118]
[0.99980905 0.99961997]
[0.99987016 0.99974158]
[0.99991171 0.99982428]
[0.99993997 0.99988052]
[0.99995918 0.99991876]
steps to convergence: 1187
expected value: [1, 1]
function result: [0.99996843 0.99993672]
```

Testing on the paraboloid $f(\mathbf{x}) = \mathbf{x}_0^2 + \mathbf{x}_1^2$ with initial guess $\begin{bmatrix}10\\12\end{bmatrix}$, tolerance 10^{-9}, initial guess $\alpha_0 = 1$ and $\alpha_1 = 1.5$:

```
Paraboloid function:
steps to convergence: 1
expected value: [0, 0]
function result: [0. 0.]
```

4 Analysis and Conclusion

The algorithm works as expected, however, the parameters are not well suited to the Rosenbrock's function shape. Thus, the secant method takes a long time to converge using our initial guess.