Math 110A Project 1

Melodye Nguyen and Austin Zhong

February 2023

1 Introduction

Steepest gradient descent is an optimization algorithm used to find the minimum of a function. The algorithm works by iteratively adjusting the parameters in the direction of steepest descent of the cost function, with the step size determined by a line search algorithm.

The secant method is a numerical method for finding the roots of a function. But in the context of SGD, the secant method is the line search algorithm, used to find the optimal step size that minimizes the cost function in the search direction.

In this report we will be applying the Steepest Gradient Descent (SGD) algorithm with Secant method in order to find the minimum of the **Rosenbrock Function**:

$$f(x, y) = (100 - x)^2 + 1(y - x^2)^2$$

using MATLAB, and will provide a mathematical overview of the implementation.

2 Code Overview

The first step is to write the code for the secant method function that will be implemented in the SGD code. The goal here is to minimize the cost function to obtain:

$$\alpha_k = argmin_{\alpha \geq 0}\ \phi_k(\alpha)$$

The secant method function takes in *phi* (cost function), $alpha0$ (initial step size guess), *tol* (stopping criteria), and returns *alpha* (optimal step size).
Next we initialize $alpha1$, $f0$, and $f1$ to obtain the two points ($alpha0$, $f0$) and ($alpha1$,$f1$). These two points are then used in the first iteration of the **secant formula**:

For $n = 1, 2, 3, \cdots$

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

In our code, $f0$, $f1$, $alpha0$, $alpha1$ are updated and the secant formula is iterated until $alpha0$ and $alpha1$ are close enough, granting us alpha, the optimal step size.
The secant formula computes a weighted average of the previous two step sizes $alpha0$ and $alpha1$, where the weights are given by the function values $f0$ and $f1$. The weight of alpha1 is proportional to the difference between $f1$ and $f0$, which ensures that the step size moves towards the minimum of the function.

Secant Method for Line Search

```
function alpha = secant(phi,alpha0,tol)
%Inputs:
    %phi = function to find optimal step size
    %alpha0 = initial alpha
    %tol = tolerance
alpha1 = alpha0 + tol;
f0 = phi(alpha0);
f1 = phi(alpha1);

%Loop the line search until covergence
while abs(alpha1 - alpha0) > tol
    %find the next approximation of the optimal alpha
    alpha2 = alpha1 - f1*(alpha1 - alpha0)/(f1 - f0);
    f2 = phi(alpha2);

    %Update new paramenters and reiterate
    alpha0 = alpha1;
    alpha1 = alpha2;
    f0 = f1;
    f1 = f2;
end
alpha = alpha2;
end
```

As mentioned before, we want to find the minimum of the Rosenbrock function. In other words, we want to find the value of the point of the function that is closest to the x-axis. First things first, we define the Rosenbrock function and the gradient of the function, as well as initialize our parameters.

The Rosenbrock function and its gradient are defined as:

$$f(\vec{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{1}$$

$$\nabla f(\vec{x}) = \begin{bmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix} \tag{2}$$

The gradient is needed to find the search direction which is a vector that points in the direction of a local minimum. In the code, the search direction is denoted as $\vec{d}$ and defined as the opposite of the gradient, $\vec{d} = -\nabla f(x)$.

Our initial parameters include N (maximum iterations) which we chose as 1000, *tol* (tolerance) which we chose as 10^{-5}, α_0 (initial step size guess) which we chose as 1, and $\vec{x}$ (initial vector) which we chose $\vec{x} = \begin{bmatrix} -1.2 \\ 1 \end{bmatrix}$. However, you can choose whatever initial parameters and the algorithm would still work as long as N and *tol* are large and small enough to reach convergence.

Then, we initialize our iteration and denote it as *iter* and run our while loop. The while loop will keep iterating until we reach the max iterations, N, or until we reach convergence. In the loop, we compute the search direction, $\vec{d}$, and then we compute the optimal step size, α, using the secant method line search mentioned earlier with our input of $\alpha_0 = 1$ and the cost function, $\phi(\alpha) := f(\vec{x}^{(k)} - \alpha \nabla f(\vec{x}^{(k)} = f(\vec{x}^{(k)} + \alpha\vec{d})$.

After finding the optimal step size, we can now use the **steepest gradient descent algorithm** to approximate the next closest $\vec{x}$. The algorithm is as follows:

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} + \alpha_k \nabla f(\vec{x}^{(k)}) = \vec{x}^{(k)} + \alpha_k \vec{d}.$$

What the algorithm is doing is taking our points ($\vec{x}$) and taking a step (α) closer in the direction of the minimum ($\vec{d}$). We check if $\|\vec{x}^{(k+1)} - \vec{x}^{(k)}\| < tol$. If so, we break the loop and $\vec{x}^{(k+1)} = \vec{x}^*$ is the minimizer. Otherwise, we update $\vec{x}$ and keep iterating.

Thus, the result of our function is that the minimum $\vec{x}^*$ is $\vec{x}^* = \begin{bmatrix} 0.996775 \\ 0.990943 \end{bmatrix} \approx \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and the minimum value is $f(\vec{x}^*) = 0.000695 \approx 0$.

Steepest Descent Algorithm

```
%Define Rosenbrock's Function
f = @(x) 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;
%Define the gradient of Rosenbrock's
grad_f = @(x) [-400*x(1)*(x(2)-x(1)^2)-2*(1-x(1));200*x(2)-x(1)^2];

%Initialize parameters
    %x = initial x vector
    %N = max iterations
    %tol = tolerance
    %alpha0 = initial alpha
x = [-1.2;1];
N = 1000;
tol = 10^(-5);
alpha0 = 1;

%Initialize iterations
iter = 0;

%Loop steepest descent algorithm until convergence and/or max iterations is
%reached
    %d = search direction
    %phi = function to find optimal step size (alpha)
    %x_new = next iteration of x
while iter < N
    d = -grad_f(x); %search direction

    phi = @(alpha) f(x+alpha*d);

    alpha = secant(phi,alpha0,tol);

    x_new = x + alpha*d;

    if norm(x_new - x) < tol
        break;
    end

    x = x_new; %update x for next iteration
    iter = iter + 1;
end
fprintf('Optimal x:(%.6f,%.6f)\n',x(1),x(2));
```

```
Minimum x:(0.996775,0.990943)
```

```
fprintf('Minimum value of Rosenbrock function: %.6f\n',f(x));
```

```
Minimum value of Rosenbrock function: 0.000695
```