Department of Mathematics and Statistics,
Queen's University

MTHE 439 Optimization Theory

Project Report: Accelerated Gradient Dynamics

Andrew Sheldon 20026539
16aws1queensu.ca

Due Friday, April 30th 2021

Contents

1 Introduction and problem formulation

This paper outlines a gradient dynamics method for solving convex optimization problems. As such, it considers a convex function f on a Hilbert space E. Moreover, f must belong to $C^{1,1}(E)$, as in for all x, $y \epsilon E$,

$$\left\| f'(x) - f'(y) \right\| \leq L \|x - y\| \tag{1}$$

This is the set of problems for which Nesterov's method applies. According to his 1983 paper [1], the method is as follows. Starting from a point $y_0 \epsilon E$, let z be an arbitrary point in E such that $z \neq y_0$ and $f'(z) \neq f'(y_0)$, and initialize the following variables.

$$k = 0$$

$$a_0 = 1$$

$$x_{-1} = y_0$$

$$\alpha_{-1} = \frac{\|y_0 - z\|}{\left\| f'(y_0) - f'(z) \right\|}$$

For each iteration k, we return the smallest i such that

$$f(y_k) - f(y_k - (\frac{1}{2})^i \alpha_{k-1} f'(y_k)) \geq (\frac{1}{2})^{i+1} \alpha_{k-1} \left\| f'(y_k) \right\|^2 \tag{2}$$

and using this i we update our variables as follows.

$$\alpha_k = (\frac{1}{2})^i \alpha_{k-1}$$

$$a_{k+1} = \frac{1}{2}(1 + \sqrt{4a_k{}^2 + 1})$$

$$x_k = y_k - \alpha_k f'(y_k)$$

$$y_{k+1} = x_k + \frac{a_k - 1}{a_{k+1}}(x_k - x_{k-1}) \tag{3}$$

The sequence of variables $\{x\}_{k=0}^{\infty}$, sometimes referred to as Nesterov's points, serves as memory terms that are factored in when updating y_k, where $\{y_k\}_{k=0}^{\infty}$ is the sequence in E upon which f is being evaluated. Within the algorithm, a_k controls the step size, and α_k serves to control the location of Nesterov's point x_k with relation to y_k.

2 Importance and application

Nesterov's gradient dynamics algorithm is best understood when compared to the accelerated gradient descent (momentum) algorithm [2]. This method is characterized by the formulas:

$$x_{k+1} = x_k - sz_k$$
$$z_k = \nabla f_k + \beta z_{k-1}$$

The momentum algorithm, also known as the heavy ball method, and the Nesterov algorithm are both accelerated descent methods, meaning they converge to the optimal point faster than the regular gradient descent algorithm. This reduces computation time, which is important in some applications and useful in all cases.

To see the similarities between the Nesterov method and the heavy ball method, particularly equation (3), observe that the Nesterov dynamics equation can be written [2] as

$$x_{k+1} = x_k + \beta(x_k - x_{k+1}) - s\nabla f(x_k + \gamma(x_k - x_{k+1}))$$

Nesterov's method is interesting and somewhat unique in that it is a gradient dynamics algorithm, not a gradient descent algorithm. That is, while it does converge to the optimal point accurately, it does not do so monotonically i.e. there is no guarantee that the sequence $\{f(y_k)\}$ in y_k is monotonically decreasing as it approaches the minimizer.

3 Theorem 1 and sketch of proof

This paper only contains two theorems, and Theorem 1 essentially provides the foundational reasons for which Nesterov's method of gradient dynamics works. Theorem 1 [1] is stated as follows.

Theorem 1 *Let $f(x)$ be a convex function in $C^{1,1}(E)$ with nonempty set of minimizers X^*. If the sequence $\{x_k\}_0^\infty$ is constructed by Nesterov's method, then the following assertions hold.*

1. *For any $k \geq 0$,*
$$f(x_k) - f^* \leq \frac{C}{(k+2)^2}$$
where $C = 4L\|y_0 - x^*\|$ and $f^* = f(x^*)$ for some $x^* \epsilon X^*$.

2. *In order to achieve accuracy within ε, the following are true.*
   (a) *the gradient of f need be computed no more than $NG = \left\lfloor \sqrt{C/\varepsilon} \right\rfloor$ times, and*
   (b) *the function f need be evaluated no more than $NF = 2NG + \lceil \log_2(2L\alpha_{-1}) \rceil$ times.*

This theorem states very little about why Nesterov's method works, and is more interested in illustrating the computation time it can guarantee.

PROOF: Note that $\{\alpha_k\}_{k=0}^{\infty}$ is a monotonically non-increasing sequence in k. Moreover, since condition (2) must be satisfied for each k, it follows that $\alpha_k \geq \frac{1}{2L}$ for each k, where α is defined by $y_k(\alpha) = y_k - \alpha f'(y_k)$ and, since $f \in C^{1,1}(E)$, by inequality (1), f satisfies

$$f(y_k) - f(y_k(\alpha)) \geq \frac{1}{2}\alpha(2 - \alpha L)\left\|f'(y_k)\right\|^2$$

Let $p_k = (\alpha_k - 1)(x_{k-1} - x_k)$. Since f is convex, the following inequalities result.

$$\langle f'(y_{k+1}), y_{k+1} - x^* \rangle \geq f(x_{k+1}) - f^* + \frac{1}{2}\alpha_{k+1}\left\|f'(y_{k+1})\right\|^2$$

$$\frac{1}{2}\alpha_{k+1}\left\|f'(y_{k+1})\right\|^2 \leq f(y_{k+1}) - f(x_{k+1}) \leq f(x_k) - f(x_{k+1}) - \frac{1}{\alpha_{k+1}}\langle f'(y_{k+1}), p_k \rangle$$

The combination of these results yields

$$\|p_{k+1} - x_{k+1} + x^*\|^2 - \|p_k - x_k + x^*\|^2 \leq 2\alpha_k a_k^2(f(x_k) - f^*) - 2\alpha_{k+1}a_{k+1}^2(f(x_{k+1}) - f^*)$$

$$2\alpha_{k+1}a_{k+1}^2(f(x_{k+1}) - f^*) \leq 2\alpha_0 a_0^2(f(x_0) - f^*) + \|p_0 - x_0 + x^*\|^2 \leq \|y_0 - x^*\|^2$$

Finally,

$$a_{k+1} = \frac{1}{2} + \frac{2\sqrt{a_k^2 + \frac{1}{4}}}{2} \geq \frac{1}{2} + a_k \geq \frac{1}{2}(k+1) + 1$$

This collection of results lays the groundwork that proves that the Nesterov programming method converges with a rate of $O(1/k^2)$ or better, i.e.

$$f(x_k) - f(x^*) \leq \frac{4L\|y_0 - x^*\|}{(k+2)^2} = \frac{C}{(k+2)^2} \leq C\frac{1}{k^2}$$

4 Application: MATLAB simulation

Consider the following function.

$$f(y_1, y_2) = c_1 y_1^2 + c_2 y_2^2$$

It can easily be verified that this function is globally convex and continuously Lipschitz in its gradient.

A MATLAB script was written to perform Nesterov's accelerated gradient dynamics method, and compare it to the gradient descent method outlined in Algorithm 9.3 of the textbook [3]. This script can be found in the Appendix. Its results are illustrated in the following figures.

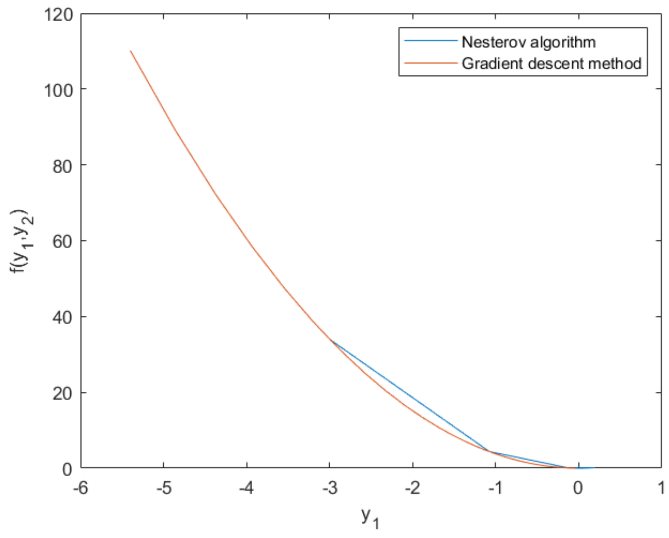

Figure 1: Simulation results for Nesterov dynamics algorithm and general gradient descent, isolated with respect to only y_1.

It is apparent from Figure 1 that the two algorithms follow roughly the same trajectory in approaching the known global minimizer at $y_1 = y_2 = 0$ of function f. This result can similarly be seen in Figure 2 where the two algorithms clearly approach the same minimizer with the same trajectory in all dimensions.

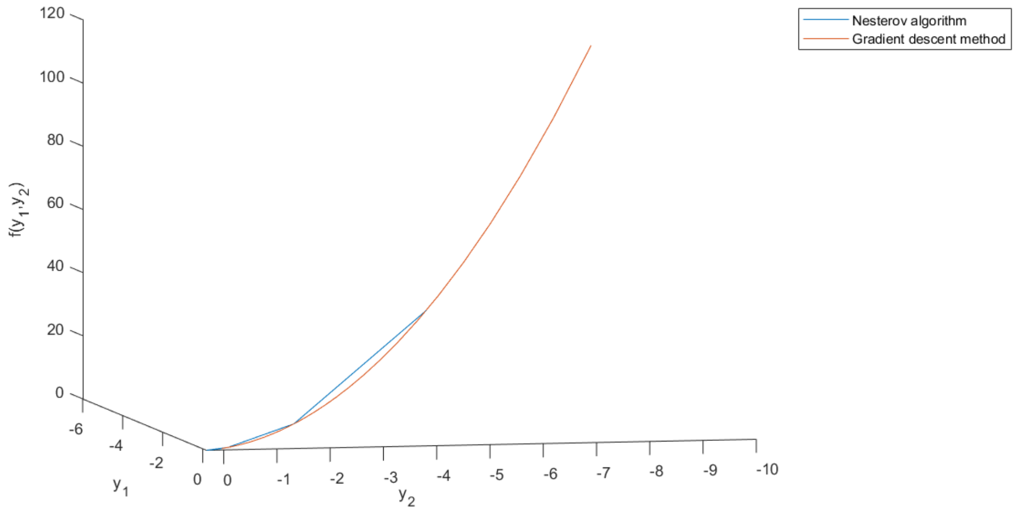

Figure 2: 3D simulation results for Nesterov dynamics algorithm and general gradient descent.

The more interesting result is shown in Figure 3. This shows the standard gradient descent algorithm requiring approximately 30 iterations to converge to the minimizer of f. Meanwhile, the Nesterov algorithm achieves a value of $f(y_1, y_2) = 0.056$ on merely its third iteration. The gradient descent algorithm requires 38 iterations to reach this same accuracy.

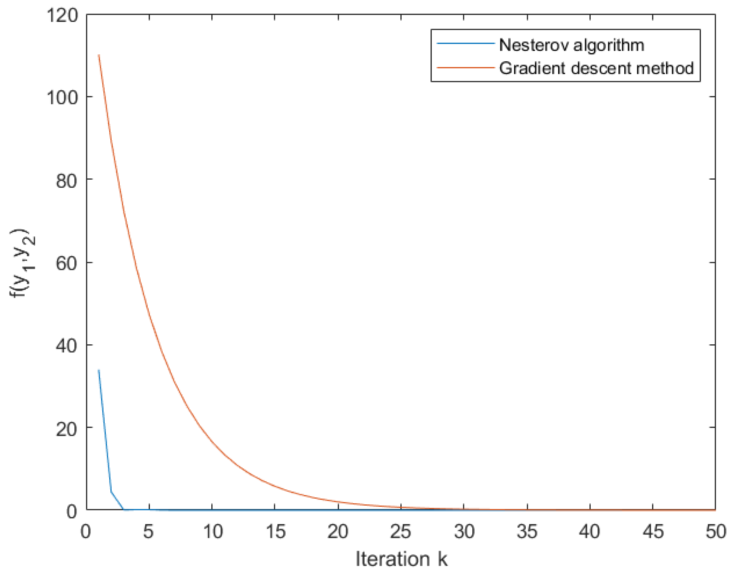

Figure 3: Value of f at each iteration of the Nesterov and gradient descent algorithms.

These values suggest that, for this function f and starting point (-6,-10), the Nesterov algorithm is 1,133.3% more computationally efficient than the gradient descent algorithm, at least with respect to the number of iterations required to reach this level of accuracy.

A Appendix: MATLAB code

A.1 nestgrad.m - Compares gradient descent simulation to Nesterov dynamics.

```
% This file aims to compare the performance of the Nesterov algorithm
% to a more standard gradient descent algorithm covered in class.

clc
clear all
close all

% initialize variables
y1 = -6;
y2 = -10;

% f = c1*y1 + c2*y2
c1 = 1;
c2 = 1;

k=0;
a=1;
x1=y1;
x2=y2;

z1=0;
z2=0;

% We start with the one-dimesional case.
y = [y1; y2];
z = [z1; z2];
alpha = norm(y-z)/norm(2*y-2*z);

% iterative steps
N=50; % may change this to a while loop with Nesterov end condition

values1 = zeros(1,N);
values2 = zeros(1,N);
values = zeros(1,N);
track1 = zeros(1,N);
track2 = zeros(1,N);

% initiate gradient descent
y_1 = y1;
```

```
y_2 = y2;      % need to track separately from Nesterov algorithm

% These track the gradient descent algorithm
track_1 = zeros(1,N);
track_2 = zeros(1,N);
values_ = zeros(1,N);
for k=1:N
    % update variables
    %f = y^2;            %f = y1^2+y2^2;
    %gradf = 2*y;        %gradf = [2*y1; 2*y2];

    i =0;
    while
             (c1*y1^2+c2*y2^2-c1*(y1-0.5^i*alpha*2*c1*y1)^2-c2*(y2-0.5^i*alpha*2*c2*y2)^2
             <= 0.5^(i+1)*alpha*norm([2*c1*y1; 2*c2*y2])^2)
        i = i+1;
    end
    %i = i+1;
    % We have our new i and can update our variables now

    alpha = 0.5^i*alpha;
    a_prev = a;
    a = 0.5*(1 + sqrt(4*a*a+1));

    x1_prev = x1;
    x2_prev = x2;

    x1 = y1 - alpha*2*c1*y1;
    x2 = y2 - alpha*2*c2*y2;

    y1 = x1 + (a_prev-1)*(x1-x1_prev)/a;
    y2 = x2 + (a_prev-1)*(x2-x2_prev)/a;

    values1(k) = y1^2;
    values2(k) = y2^2;
    values(k) = c1*y1^2 + c2*y2^2;
    track1(k) = y1;
    track2(k) = y2;

    % Perform gradient descent:

    del_y1 = -2*c1*y_1;    % negative gradient of f at y1,y2
    del_y2 = -2*c2*y_2;
    t=0;
    while (c1*(y_1+t)^2 + c2*(y_2+t)^2 < c1*y_1^2 + c2*y_2^2)
```

```
        t = t+0.05;
    end
    t = t+0.05;
    % We should now have (approximately) optimal t
    % Now we update our y values (descent method step 3)
    y_1 = y_1 + t*del_y1;
    y_2 = y_2 + t*del_y2;

    % Track gradient descent
    track_1(k) = y_1;
    track_2(k) = y_2;
    values_(k) = c1*y_1^2 + c2*y_2^2;
end

l = 1:N;
%plot(l,track1)

%figure(2)
%plot(l,track2)

%figure(3)
%plot(track1, track2)

figure(4)
plot3(track1, track2, values)
hold on
plot3(track_1, track_2, values_)
hold off

%figure(5)
%plot(l, values-values_)

figure(6)
%plot(track1, values)
plot(l, values)
hold on
plot(l, values_)
%plot(track_1, values_)
%
hold off
figure(7)
plot(track1, values)
hold on
plot(track_1, values_)
```

References

[1] Yuri Nesterov. "A method of solving a convex programming problem with convergence rate O($1/k^2$)". In: *Soviet Math. Dokl.* 27.2 (1983).

[2] Gilbert Strang. *23. Accelerating Gradient Descent (Use Momentum)*. MIT OpenCourseWare. 2019.

[3] Steven Boyd; Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.