SOLUTION 9 – Advanced Numerical Analysis

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Prof. D. Kressner
M. Steinlechner

1 ▶ Accelerated Gradient Descent

a) We consider the overdetermined linear system of equations $A\mathbf{x}^* + \mathbf{w} = \mathbf{b}$ with $A \in \mathbb{R}^{n\times p}$ a matrix with full column rank, $\mathbf{x} \in \mathbb{R}^p$ the unknown solution, and $\mathbf{w} \in \mathbb{R}^n$ denotes unknown noise. The least squares estimator for the unknown solution is then given by

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x}), \quad \text{with} \quad f(\mathbf{x}) := \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^p} \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|_2^2.$$

Show that f is a μ-strongly convex function with μ given by the smallest eigenvalue of $A^\mathsf{T}A$. Furthermore, show that the gradient of f is Lipschitz continuous with the smallest Lipschitz constant L given by the largest eigenvalue of $A^\mathsf{T}A$.

We obtain the Hessian $H(x) = A^\mathsf{T}A$, hence f is μ-strongly convex with

$$\mu = \lambda_{min}(A^\mathsf{T}A),$$

and the gradient of f is Lipschitz continuous with Lipschitz constant

$$L = \lambda_{max}(A^\mathsf{T}A).$$

b) Implement Algorithm 4.28, *Accelerated Gradient Descent*. Use the provided template on the lecture homepage, `AGD.m`

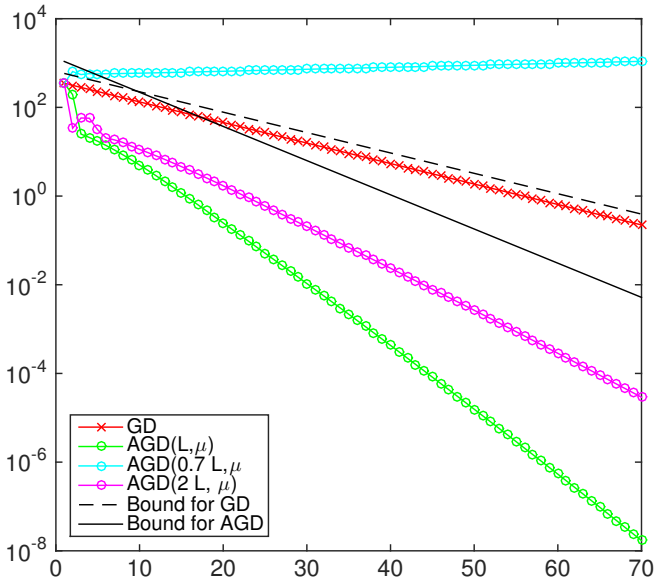
```
function [X,fX] = AGD( f, df, x0, mu, L, opts );
    % load default values for option parameters if not set by user
    if ~exist( 'opts', 'var');     opts = struct();       end
    if ~isfield( opts, 'tol');     opts.tol = 1e-6;       end
    if ~isfield( opts, 'maxiter'); opts.maxiter = 1000;   end

    X(:,1) = x0;
    fX(:,1) = f(x0);

    xk = x0;
    yk = x0;
    gk = df(x0);

    kappa = sqrt(mu / L);
    kappa = (1 - kappa) / (1 + kappa);

    k = 1;
    while norm(gk) > opts.tol && k < opts.maxiter
        % perform step
        yk_new = xk - gk / L ;
        xk = yk_new + kappa * (yk_new - yk);
        yk = yk_new;

        % prepare next step
        gk = df(xk);
        X(:,k+1) = xk;
        fX(:,k+1) = f(xk);

        k = k + 1;
    end

end
```

c) We will now investigate the convergence rates of standard Gradient Descent (GD) and Accelerated Gradient Descent (AGD) when applied to the cost function f. An implementation of GD is available on the lecture homepage, `GD.m`. Using

the provided template `ex9problem1.m`, plot the convergence rates of GD for the optimal choice $\alpha = \frac{2}{L+\mu}$ which exploits the strong convexity of the function. Compare to the convergence rate of AGD using your implementation from **b)**. Try out how the convergence speed changes as you use non-optimal values for L and μ.

```
rng(11);

% set up least squares problem
n = 500;
p = 10;
A = rand( n, p );
x = rand( p, 1 );
noise = 1e-5*randn( n, 1 );
b = A*x + noise;

% obtain the exact minimizer by direct least squares solver (as the noise is gaussian)
% Note that due to the noise, x is not the same as x_exact
x_exact = A\b;

% Definition of function, derivative and Hessian
f = @(x) 0.5 * norm( A*x - b )^2;
df = @(x) A' * (A*x - b);
H = A'*A;

fx_exact = f(x_exact);

ev = eig(H);

% Compute Lipschitz constant L and strong convexity parameter mu exactly
L = max(ev)
mu = min(ev);

opts.maxiter = 70;
opts.tol = 1e-6;

x0 = rand( p, 1 );

[X_GD,  fX_GD ]       = GD ( f, df, x0, 2/(L+mu) , opts );
[X_AGD, fX_AGD]       = AGD( f, df, x0, mu, L    , opts );
% Run AGD again with "wrong" constants
[X_AGD_07, fX_AGD_07] = AGD( f, df, x0, mu, 0.7*L, opts );
[X_AGD_2, fX_AGD_2]   = AGD( f, df, x0, mu, 2*L  , opts );

% Check for convergence:
norm( x_exact - X_GD(:,end) )
norm( x_exact - X_AGD(:,end) )
norm( x_exact - X_AGD_07(:,end) )
norm( x_exact - X_AGD_2(:,end) )


semilogy( 1:length(fX_GD), fX_GD - fx_exact, '-xr' )
hold on
semilogy( 1:length(fX_AGD),    fX_AGD - fx_exact,    '-og' )
semilogy( 1:length(fX_AGD_07), fX_AGD_07 - fx_exact, '-oc' )
semilogy( 1:length(fX_AGD_2) , fX_AGD_2 - fx_exact,  '-om' )

set(gca,'fontsize',16)

k = 1:opts.maxiter;
kappa = L / mu;
gamma = (kappa-1) / (kappa+1);
semilogy( k, 0.5*L*gamma.^(2*k), '--k')
semilogy( k, L*(1-sqrt(mu/L)).^(k), '-k')

l = legend('GD', 'AGD(L,\mu)', 'AGD(0.7 L,\mu', 'AGD(2 L, \mu)', 'Bound for GD', 'Bound for AGD')
set(l,'location','SouthWest')

```

```
64 | hold off
```



2 ▶ Binary logistic regression, part 2

In the last exercise sheet, we have seen the *binary logistic regression* approach. Let $\mathbf{a}_i \in \mathbb{R}^n$ be sampling points and b_i be the associated binary class labels. Then, we want to determine the maximum log-likelyhood estimator

$$\operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x}), \qquad \text{with } f(\mathbf{x}) = -\sum_{i=1}^{n} \log\left(h(b_i \mathbf{a}_i^\mathsf{T}\mathbf{x})\right),$$

where $h(t) = 1/(1+\exp(t))$ is the sigmoid function. In the last exercise sheet, we have shown that the function is convex, but in general not strongly convex, depending on the data matrix $A = \begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n\end{bmatrix}^\mathsf{T}$.

Hence, we introduce a regularization term, such that the minimization problem is now given as

$$x^* = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^p} f_\sigma(\mathbf{x}), \qquad \text{with } f_\sigma(\mathbf{x}) = f(\mathbf{x}) + \frac{\sigma}{2}\|x\|_2^2.$$

a) Show that the regularized function f_σ is σ-strongly convex.

The Hessian of the regularized function f_σ is given by

$$H_\sigma = H(x) + \sigma I,$$

where $H(x)$ is the Hessian of the non-regularized function f with which we worked in the last exercise. There, it was shown to be positive semi-definite. Hence, if $H(x)$ has eigenvalues λ_i, then H_σ has the shifted eigenvalues $\lambda_i + \sigma$. Therefore, the f_σ μ-strongly convex with parameter $\mu = \sigma$.

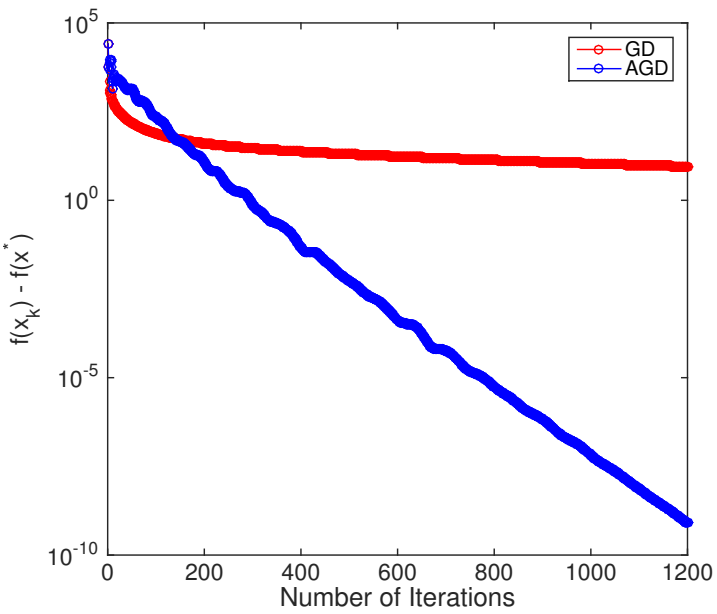
b) Show that the gradient of f_σ is Lipschitz continuous with $L = \frac{1}{4}\|A\|_2 + \sigma$.

*With the same argument as in **a)**, the maximum eigenvalue of $H_\sigma(x)$ is the maximum eigenvalue of the non-regularized Hessian $H(x)$ shifted by the regularization parameter σ.*

c) We will now test the performance of GD and AGD applied to binary logistic regression. For simplicity, we choose the regulatization parameter $\sigma = 1$. As a dataset, we will use the `a4a` dataset from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html, a processed version of the "Adult" dataset. This dataset aims to predict from 4781 samples from the1994 US census the probability of a person to earn over 50000 USD per year given 123 attributes

such as age, sex, education, etc. Complete the template `ex9problem2.m` using your implementation of AGD from the previous question and the prepared dataset `adult.mat` from the lecture homepage.

```
% Load the dataset to get matrices A, b and the exact
% solution xexact
load('adult.mat')

% Regularization parameter
sigma = 1;

% Lipschitz constant
L = 0.25*svds(A,1)^2 + sigma;

% strong convexity parameter
mu = sigma;

h = @(t) 1./(1 + exp(-t));
f = @(x) -sum(log(h(b.*(A*x)))) + 0.5*sigma*x'*x;
df = @(x) -A' * ( (1-h(b.*(A*x))) .* b ) + sigma*x;

fxexact = f(xexact)

opts.maxiter = 1200;
opts.tol = 1e-6;

x0 = rand( size(A,2), 1 );

[X_GD,  fX_GD ] = GD ( f, df, x0, 2/(L+mu) , opts );
[X_AGD, fX_AGD] = AGD( f, df, x0, mu, L    , opts );


semilogy( 1:length(fX_GD), fX_GD - fxexact, '-or' )
hold on
set(gca,'fontsize',16)
semilogy( 1:length(fX_AGD),fX_AGD - fxexact,'-ob' )
xlabel('Number of Iterations')
ylabel('f(x_k) - f(x^*)')

legend('GD','AGD')
```



3 ▶ Tangent cones

Follows directly from the definition. See also discussion in the exercise class.