# Homework 1 : gradient descent and convexity

Noé Hollande, Thomas Renard, Yujun Kim

22 mars 2022

## 1 Theoretical Questions

Question 1 :

We have the following transformation : f($\theta$) = -log (l($\theta$))
= -log( $\prod_{i=1}^{m} \sigma(\langle x_i, x\rangle + b)^{y_i}\sigma(-\langle x_i, x\rangle - b)^{1-y_i} = -\sum_{i=1}^{m} log(\sigma(\langle x_i, x\rangle + b)^{y_i}) + log(\sigma(-\langle x_i, x\rangle - b)^{1-y_i})$.
Using $-log(\sigma(x)) = log(1/\sigma(x)), log(x^y) = ylog(x)$ and the indicated notation $\tilde{x}_i = (x_i, 1)$, we get the desired expression : f($\theta$) = $\sum_{i=1}^{m} y_i log(1 + e^{-\langle \tilde{x}_i, \theta\rangle}) + (1 - y_i)log(1 + e^{\langle \tilde{x}_i, \theta\rangle})$.

Question 2 :

We compute the second derivative of f(z) = $log(1 + e^z)$.
$f'(z) = e^z/1 + e^z$
$f''(z) = e^z(1 + e^z) - e^{2z}/(1 + e^z)^2 = e^z/(1 + e^z)^2$ which is always positive since the upper and the lower terms are, by Analysis I we know this proves that f(z) is convex.

Question 3 :

$a(\theta) = log(1+e^{(-)\langle \tilde{x}_i, \theta\rangle})$ is the composition of the function $f(z) = log(1+e^z)$ by $g(\theta) = (-)\langle \tilde{x}_i, \theta\rangle$.

$g(\theta)$ is convex because it's linear (so the second derivative equals 0) and $f(z)$ has be proven to be convex previously, plus it's non decreasing (because $e^z$ is non decreasing and neither is log(x)), by Exercise 4.20 in the poly, $a(\theta)$ is convex (note that the (-) in the previous lines were here to signify that this is true for both $\langle \tilde{x}_i, \theta\rangle$ and $-\langle \tilde{x}_i, \theta\rangle$.

$f(\theta)$ is an sum of convex functions multiplied by positive coefficients (all the $y_i$ and $(1 - y_i)$ are positive since $y_i$ is 0 or 1). By Exercice 4.13 in the poly, this proves the convexity of f.

Question 4 :

By definition of strong convexity $f_\lambda$ is strongly convex if $f_\lambda(\theta) - \delta/2\|\theta\|^2$ is convex for some parameter $\delta > 0$, here we can conveniently take $\delta = \lambda$ the parameter of $f_\lambda$ to have $f_\lambda(\theta) - \lambda/2\|\theta\|^2 = f(\theta)$ which we proved was convex in the previous question .

Question 5 :

First, let's argue that $f_\lambda$ has at most one minimum : Suppose $\exists$ x,y such that $f\lambda$ reaches its minimum on both points : we'd have, for all t $\in (0,1)$ :

$$min \leq f((1-t)x + ty) < (1-t)f(x) + tf(y) = min$$

Where the first inequality holds because min is the minimum of the fuction and the second because $f_\lambda$ is stricly convex. Hence a contradiction.

The existence of this minimum is proven in Theorem 4.27, the fact that it's on a point $\theta$ such that $_\lambda(\theta) = 0$ is obvious (as it's a necessary condition for a point to be a minimum). The second derivative is positive definite because the function is strongly convex, hence the derivative is strictly increasing, proving there can be at most one value where it cancels itself and we're done.

Question 6 :

$\nabla f_\lambda(\theta) = \nabla f(\theta) + \lambda\theta$.

Define $\phi(\theta) = log(1 + e^{\langle x,\theta\rangle})$
Hence : $\nabla\phi(\theta) = e^{\langle x,\theta\rangle}/1 + e^{\langle x,\theta\rangle} * x$.

Hence we have $\nabla f_\lambda(\theta) = \sum_{i=1}^m y_i(e^{\langle -x_i,\theta\rangle}/1 + e^{\langle -x_i,\theta\rangle}) * (_{xi}) + \sum_{i=1}^m (1-y_i)(e^{\langle x,\theta\rangle}/1 + e^{\langle x,\theta\rangle}) * x_i + \lambda\theta$.

# 2 Classical Gradient Descent

The `main.m` code handle overall process. We set the parameter `lambda`. `fhandle` takes training data, theta, and lamda and produce the function value with the gradient at given point. We check that the first order *Taylor Expansion* has $O(t^2)$ error. Next, we calculate the step size for gradient descent, and implement gradient descent. The norm of gradient at each iteration is then saved as *png* file.

```
%%  Define the cost function f (question 7)
lambda = 1e-4;
fhandle = @(theta) logistic_regression(train, theta, lambda);

%% Check the gradient (question 8)
theta = randn(d, 1);
v = randn(d, 1);
v = v/norm(v);
checkgradient(fhandle, theta, v);

%% Lipschitz constant (question 9)
X = train.X;
L = 1/4 * norm(X)^2 + lambda;

%% Gradient descent with constant step sizes (question 10)
params.maxiters = 30000;
params.maxtime = 3 * 60;
```

```
18 params.tolgradnorm = 5e-4;
19 params.verbose = false;
20 alpha = 1/L;
21 theta0 = rand(d, 1);
22 [theta, gradnorms, times] = cstgradientdescent(fhandle, theta0, alpha, ...
       params);
23
24 %% Plot results (question 11)
25 semilogy(gradnorms);
26 saveas(gcf, 'gradnorms.png')
```

Listing 1: main.m

## 2.1 Problem 7

The function `logistic_regression` gets train data, theta, and lambda to calculate the function value and its gradient for logistic regression. To avoid *NaN* problem, code was implemented in care.

```
1  function [f g] = logistic_regression(train, theta, lambda)
2      X = train.X;
3      y = train.y;
4      dots = X.'*theta;
5      lg = @(x) log(1 + exp(x));
6      f_theta = dot(y,lg(-dots)) + dot((ones(length(y),1)) -y,lg(dots));
7
8      f = f_theta + (lambda/2) * norm(theta)^2;
9
10     % computation of g
11     sigmoid = @(x) exp(x)./(1+exp(x));
12     g = X * (-y.*sigmoid(-dots)+(ones(length(y),1)-y).*sigmoid(dots)) + ...
           lambda * theta;
13
14
15
16     if any(isnan(f), 'all') || any(isnan(g), 'all')
17         warning(['The function logistic_regression returned expressions ...
               containing NaNs.' ...
18             'Read the section numerical considerations at the end of the ...
                  homework and debug your code.']);
19     end
20 end
```

## 2.2 Problem 8

As seen on figure 1, the slope of error with respect to variable $t$ equals 2(It has same slope as $t^2$). The Thus, we can deduce that the code. `checkgradient` is reasonable.

```
1  function checkgradient(fhandle, theta, v)
2  % Numerically check if the gradient is correct.
3
4  % fhandle: Function handle returning the function value and the gradient at
5  % a given point.
6  % theta: Point where to perform the Taylor expansion.
```

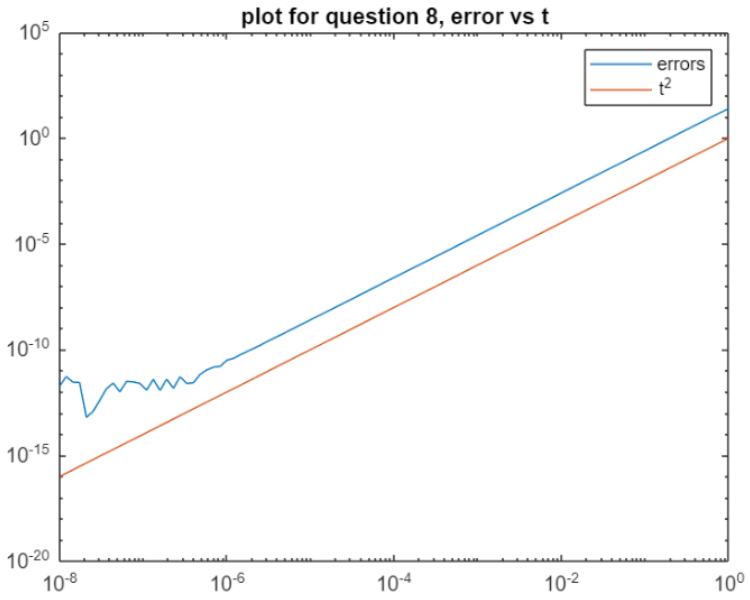


FIGURE 1: Error of 1st Order Taylor Expansion(Log-Log Scaled)

```
% v: Direction in which to perform the Taylor expansion.
    n = 101;
    ts = logspace(-8, 0, n);

    %Calculate f and grad of f
    [f0, g0] = fhandle(theta);
    f0 = f0*ones(1,n);
    dots = dot(v,g0);
    dots = ts*dots;
    A = repmat(v,1,n);
    A = ts.*A;
    A = theta + A;

fs = zeros(1,n);
    for i = 1:n
        fs(i) = fhandle(A(:,i));
    end

    %calculate errors

        errors = abs(fs - f0 - dots);


    %plotting
    slope = @(x) x.^2;
    h = slope(ts);

    loglog(ts, errors)
    hold on
    loglog(ts,h)
    hold off

    title('plot for question 8, error vs t')
    legend('errors', 't^2')

end
```

## 2.3 Problem 9

Every code is in `main.m`.

## 2.4 Problem 10

```
function [theta, gradnorms, times] = cstgradientdescent(fhandle, x0, ...
    alpha, params)
    maxiters = params.maxiters;
    maxtime = params.maxtime;
    tolgradnorm = params.tolgradnorm;

    gradnorms = zeros(1, maxiters + 1);
    times = zeros(1, maxiters + 1);
    iter = 0;
    xk = x0;
    [f0 g0] = fhandle(x0);
    init_grad_norm = norm(g0, 2);

    tic
    while (iter < maxiters && toc < maxtime)
        iter = iter + 1;
        [f, g] = fhandle(xk);
        xk = xk - alpha*g;
        grad_norm = norm(g, 2);
        gradnorms(iter) = grad_norm;
        times(iter) = toc;
        if rem(iter,100) == 0
            iter
            toc
            grad_rate = grad_norm/init_grad_norm
        end
        if grad_norm < init_grad_norm * tolgradnorm
            break
        end
    end

    theta = xk;
    gradnorms = gradnorms(1:iter);
    times = times(1:iter);
end
```

## 2.5 Problem 11

Figure 2 shows the norm of gradients at each iteration. As the point theta limits to minimum, the norm of gradients decrease.

## 2.6 Problem 12

Recall a theorem from the lecture.

**Theorem** (4.31). *Assume $f : \xi \longrightarrow \mathbb{R}$ is $\lambda$ strongly convex and has $L$-Lipschitz gradient. Let $x_0 \in \xi$ be arbitary. Run GD as follows :*

$$x_{k+1} = x_k - 1/L\nabla f(x_k)$$

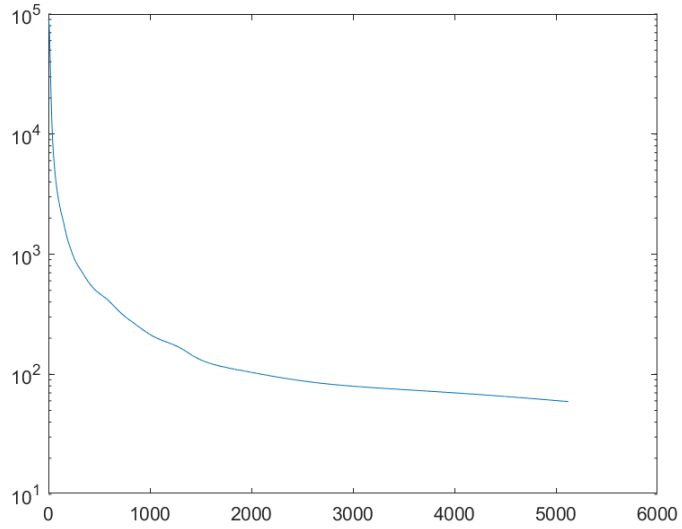


FIGURE 2: Norm of Gradients at Each Iteration

*Then, there is unique minimizer $x^*$ of $f$ and*

$$f(x_k) - f(x^*) \leq \left(1 - \frac{1}{\kappa}\right)^k (f(x_k) - f(x^*)) \cdots (*)$$

*where* $\kappa = \frac{L}{\lambda}$

In our situation, the function $f_\lambda$ satisfies given conditions. It however, remains to show $\nabla f$ is $L = \sigma(X)^2_{max}/4 + \lambda$ Lipschitz continuous. This can be shown as following : Let $\theta, \phi \in \xi$. Note that $\nabla f(\theta) = X(s(X^t\theta) - y) + \lambda\theta$, where $s : \mathbb{R}^m \longrightarrow \mathbb{R}^m$ is element-wise applying sigmoid function. Note that sigmoid function $sig : \mathbb{R} \longrightarrow \mathbb{R}$ is 1/4-Lipschitz continuous. This is proved by calculating its derivative.

$$sig'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{2 + e^x + e^{-x}} \leq \frac{1}{2+2} = \frac{1}{4}$$

Then, $s$ is also 1/4-Lipschitz continuous. If $\theta = (x_1, \cdots, x_m), \phi = (y_1, \cdots, y_m)$, then

$$|s(\theta) - s(\phi)|^2 = \sum (sig(x_i) - sig(y_i))^2 \leq \sum ((x_i - y_i)/4)^2 = (|\theta - |/4)^2$$

and so $|s(\theta) - s(\phi)| \leq |\theta - \phi|/4$. Now, $\nabla f$ is Lipschitz continuous by following argument.

$$\begin{aligned}
|\nabla f(\theta) - \nabla f(\phi)| =& |X(s(X^T\theta)) - X(s(X^T\phi)) + \lambda(\theta - \phi)| && (1)\\
\leq& |\sigma(X)_{max}||s(X^T\theta) - s(X^T\phi)| + \lambda|\theta - \phi| && (2)\\
\leq& \frac{|\sigma(X)_{max}|}{4}|X^T\theta - X^T\phi| + \lambda|\theta - \phi| && (3)\\
\leq& \frac{|\sigma(X)_{max}|^2}{4}|\theta - \phi| + \lambda|\theta - \phi| && (4)\\
=& \left(\frac{|\sigma(X)_{max}|^2}{4}| + \lambda\right)|\theta - \phi| && (5)
\end{aligned}$$

Hence by applying, *Corollary 4.32* in the lecture note, we conclude that $x_k$ found above converges to the unique minimizer $x^*$ at least linearly.

## Comments Through Problem 7 - Problem 12

The two plotting questions above was shown as expected. The error of first order Taylor expansion was $O(t^2)$ by showing slope of 2 on log-log plotting. Also, norm of Gradients at each iteration was decreasing. As norm of gradient decrease linearly, the log scale graph should look linear, which actually is eventually from iteration 2000+.

# 3 Line Search Gradient Descent

The following code is `main.m` for LS GD.

```
%Problem 13
% Parameters for the main loop
params.maxiters = 30000;
params.maxtime = 3*60;
params.tolgradnorm = 1e-5;
params.verbose = false;

% Examples of parameters for the linesearch
lsparams.alphabar = 10^-4;
lsparams.c = 1e-4;
lsparams.rho = 0.7;
lsparams.alphamin = 1e-8;

% Initial point
% theta0 = ...
thetals0 = rand(d, 1);

% Run gradient descent with linesearch
% ... lsgradientdescent(fhandle, theta0, params, lsparams); ...
[thetals, gradnormsls, timesls] = lsgradientdescent(fhandle, thetals0, ...
    params, lsparams)
```

This is code for `lsgradient.m`.

## 3.1 Problem 13

```
function [theta, gradnorms, times] = lsgradientdescent(fhandle, x0, ...
    params, lsparams)
    maxiters = params.maxiters;
    maxtime = params.maxtime;
    tolgradnorm = params.tolgradnorm;

    gradnorms = zeros(1, maxiters + 1);
    times = zeros(1, maxiters + 1);
    iter = 0;
    xk = x0;
    alpha = linesearch(fhandle, x0, lsparams);
    [f0, g0] = fhandle(x0);
    init_grad_norm = norm(g0, 2);

    tic
    while (iter < maxiters && toc < maxtime)
        iter = iter + 1;
```

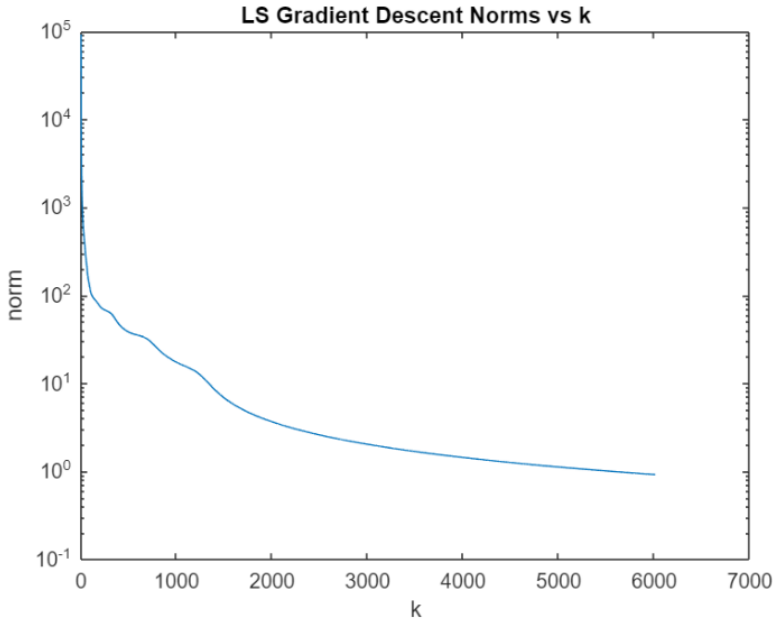


FIGURE 3: Problem 13 LS Gradient Norm

```
        [f, g] = fhandle(xk);
        xk = xk - alpha*g;
        alpha = linesearch(fhandle, xk, lsparams);
        grad_norm = norm(g, 2);
        gradnorms(iter) = grad_norm;
        times(iter) = toc;
        if rem(iter,100) == 0
            iter
            toc
            grad_rate = grad_norm/init_grad_norm
        end
        if grad_norm < init_grad_norm * tolgradnorm
            break
        end
    end

    theta = xk;
    gradnorms = gradnorms(1:iter);
    times = times(1:iter);
end
```

## 3.2 Problem 14

As shown below, *LS gradient* has steeper descent for the norm of gradient.

## 3.3 Problem 15

$b$ is the last element of calculated $\theta$, and it is $b = -0.2840$. The image of theta is shown as figure 15.

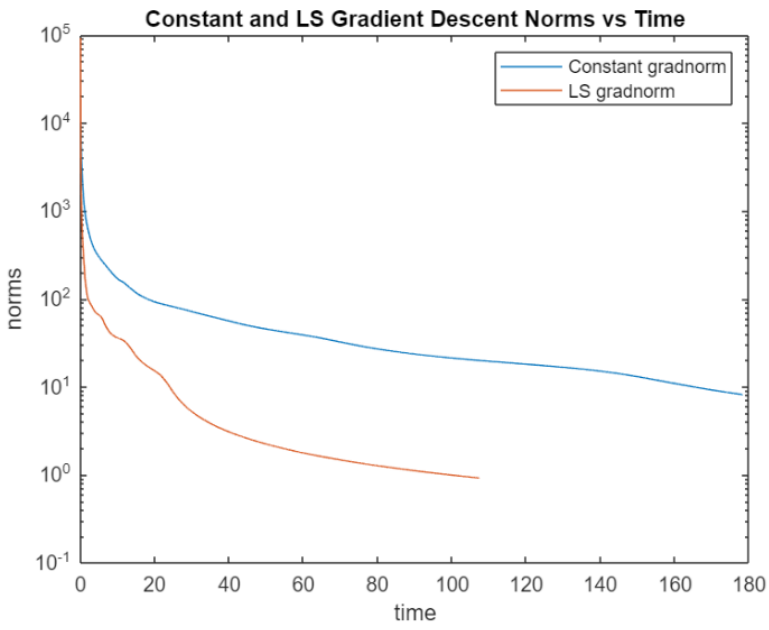


FIGURE 4: Problem 14 Norm Compare

## 3.4 Problem 16

The accuracy was
Train : 1
Test : 0.999527
The failed test image is shown as figure 6.

As the rate of correct guess for the test cases shows, the theta we have found works well. The only failed test image indeed failed actually looks confusing, thus we conclude that logistic regression found by gradient descent for binary image classification problem between 0 and 1 works well.

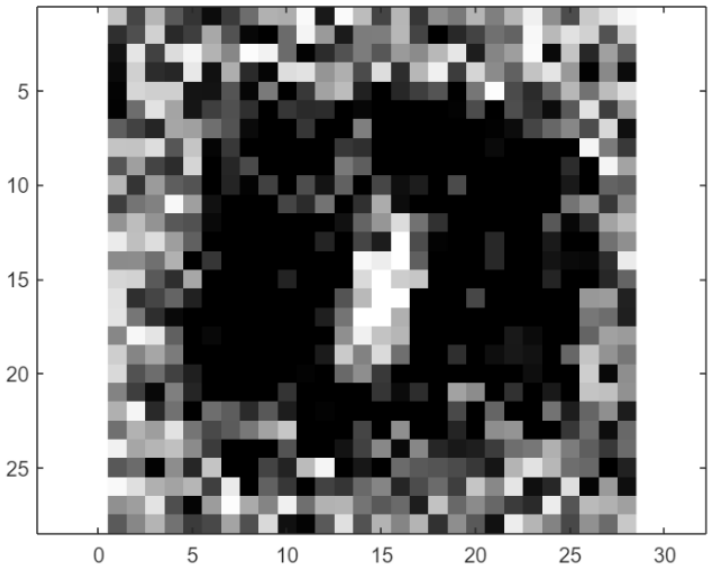


FIGURE 5: Problem 15 Theta

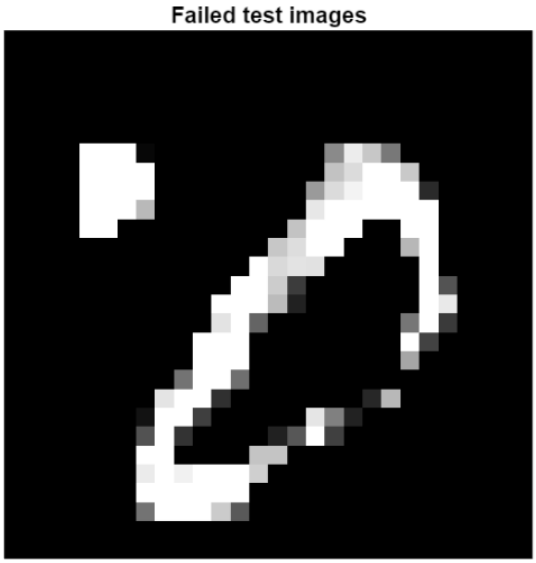


FIGURE 6: Problem 16 Failed Image