



THE UNIVERSITY OF TEXAS  
AT AUSTIN

---

EE381V LARGE SCALE OPTIMIZATION

**Problem Set 1**

Edited by L<sup>A</sup>T<sub>E</sub>X

Department of Computer Science

---

STUDENT

**Jimmy Lin**

xl5224

COURSE COORDINATOR

**Sujay Sanghavi**

UNIQUE NUMBER

**17350**

RELEASE DATE

**September 12, 2014**

DUE DATE

**September 18, 2014**

TIME SPENT

**10 hours**

September 15, 2014

## Table of Contents

<b>I</b>	<b>Matlab and Computational Assignment</b>	<b>2</b>
<b>1</b>	<b>Gradient Descent on three matrices</b>	<b>2</b>
1.1	$X1, b1$ . . . . .	2
1.2	$X2, b2$ . . . . .	3
1.3	$X3, b3$ . . . . .	4
<b>2</b>	<b><math>\gamma = 1</math> for the second matrix</b>	<b>5</b>
<b>II</b>	<b>Written Problems</b>	<b>6</b>
<b>1</b>	<b>Othorognal Subspace</b>	<b>6</b>
(a)	Show that if $U$ is a subspace, then so is $U^\perp$ . . . . .	6
(b)	Show that $(U^\perp)^\perp = U$ . . . . .	6
(c)	Show that if $U, W \subseteq V$ are subspaces of $V$ , then $U \subseteq W \Leftrightarrow U^\perp \supseteq W^\perp$ . . . . .	6
(d)	Show that $X^\perp$ makes sense, $X^\perp$ is a subspace and $(X^\perp)^\perp \supseteq X$ . . . . .	6
(e)	Show that any $v \in V$ can be written uniquely as $v = u + u^\perp$ . . . . .	6
<b>2</b>	<b>Boyd and Vandenberghe, Ex. 2.10</b>	<b>7</b>
(a)	Show that if $A \in \mathbb{S}_+^n$ then the set $C$ is convex . . . . .	7
(b)	Show that $C_1$ is convex if there exists $\lambda \in \mathbb{R}$ such that $(A + \lambda gg^T) \in \mathbb{S}_+^n$ . . . . .	7
<b>3</b>	<b>Boyd and Vandenberghe, Ex. 2.21</b>	<b>8</b>
<b>7</b>	<b>Form a Half-Space</b>	<b>9</b>
<b>8</b>	<b>Exists <math>C</math> such that <math>CA = B</math></b>	<b>10</b>
<b>A</b>	<b>Codes Printout</b>	<b>11</b>
(a)	Gradient Descent Routine . . . . .	11
(b)	Running Script . . . . .	12

## List of Figures

1	Illustration for gradient descent on $X1$ , staring with $b1$ by $\gamma = 0.5$ and $3.0$ . . . . .	2
2	Illustration for gradient descent on $X2$ , starting with $b2$ by $\gamma = 1.5$ and $3.0$ . . . . .	3
3	Illustration for gradient descent on $X3$ staring with $b3$ by $\gamma = 0.005$ and $0.05$ . . . . .	4
4	Plotting figure for gradient descent with $\gamma = 1$ on the second matrix . . . . .	5

## Part I

## Matlab and Computational Assignment

## 1 Gradient Descent on three matrices

Command to get executed:

```
>> gd_run_script()
```

1.1  $X1, b1$ 

- Range of  $\gamma$  that leads to convergence:  $(0, 2)$
- Range of  $\gamma$  that leads to divergence:  $(2, +\infty)$
- Explanation: if  $\gamma = 2$ , the program indicates that

$$\forall k, f(x^{k+1}) = f(x^k)$$

Since the above equation is constantly true (independent of the minima), we can conclude that gradient descent with  $\gamma = 2$  goes to the opposite side of that quadratic curve. Intuitively, the program will diverge if we set larger ( $\gamma > 2$ ) and converge if we set smaller ( $\gamma < 2$ ).

- Two illustrative examples:  $\gamma = 0.5$  and  $\gamma = 3.0$

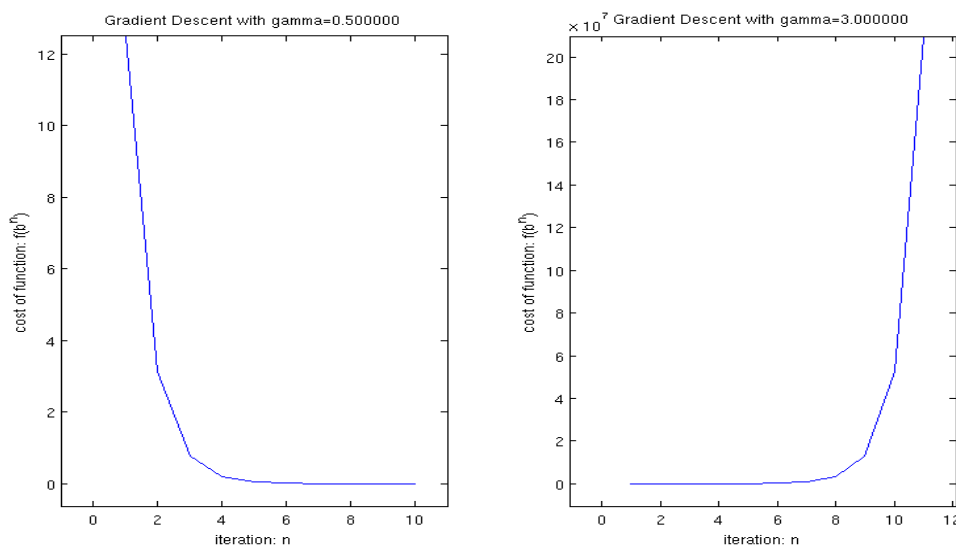


Figure 1: Illustration for gradient descent on  $X1$ , starting with  $b1$  by  $\gamma = 0.5$  and  $3.0$

## 1.2 $X2, b2$

- Range of  $\gamma$  that leads to convergence:  $(0, 2)$
- Range of  $\gamma$  that leads to divergence:  $(2, +\infty)$
- Explanation: if  $\gamma = 2$ , the program indicates that

$$\forall k, f(x^{k+1}) = f(x^k)$$

Since the above equation is constantly true (independent of the minima), we can conclude that gradient descent with  $\gamma = 2$  goes to the opposite side of that quadratic curve. Intuitively, the program will diverge if we set larger ( $\gamma > 2$ ) and converge if we set smaller ( $\gamma < 2$ ).

- Two illustrative examples:  $\gamma = 1.5$  and  $\gamma = 3.0$

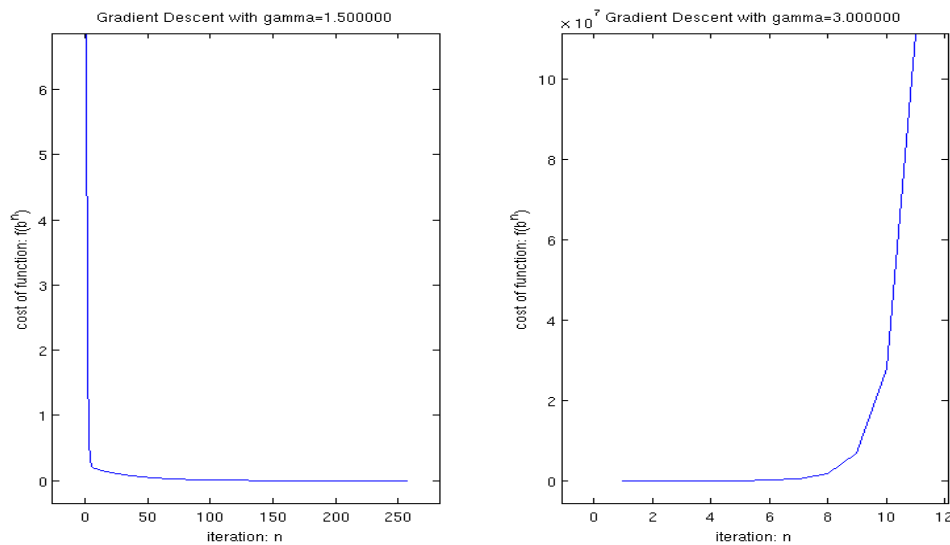


Figure 2: Illustration for gradient descent on  $X2$ , starting with  $b2$  by  $\gamma = 1.5$  and  $3.0$

### 1.3 $X3, b3$

- Range of  $\gamma$  that leads to convergence:  $(0, 0.02)$
- Range of  $\gamma$  that leads to divergence:  $(0.02, +\infty)$
- Explanation: if  $\gamma = 0.02$ , the program indicates that

$$\forall k, f(x^{k+1}) = f(x^k)$$

Since the above equation is constantly true (independent of the minima), we can conclude that gradient descent with  $\gamma = 0.02$  goes to the opposite side of that quadratic curve. Intuitively, the program will diverge if we set larger ( $\gamma > 0.02$ ) and converge if we set smaller ( $\gamma < 0.02$ ).

- Two illustrative examples:  $\gamma = 0.005$  and  $\gamma = 0.05$

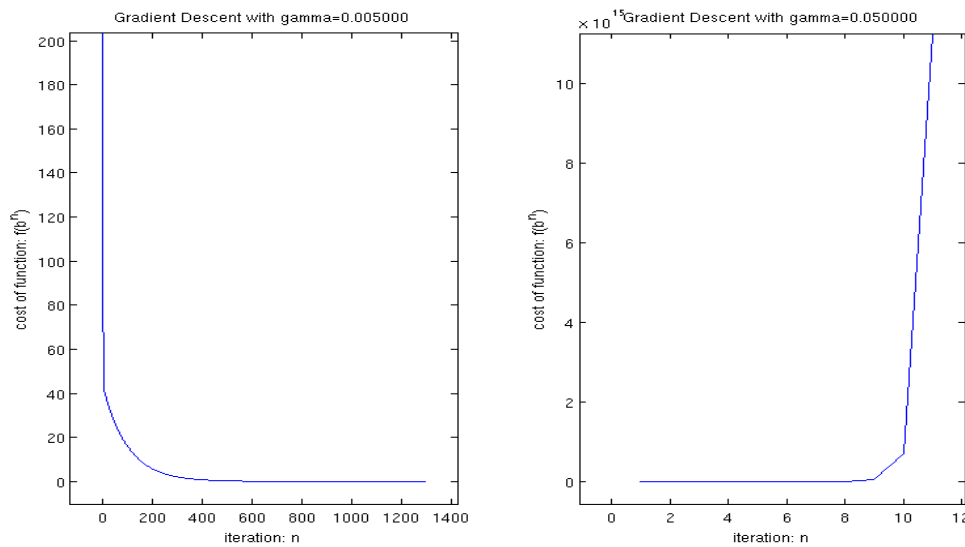


Figure 3: Illustration for gradient descent on  $X3$  starting with  $b3$  by  $\gamma = 0.005$  and  $0.05$

## 2 $\gamma = 1$ for the second matrix

Command to get executed:

```
>> [b2_opt, iters, all_costs] = gd (X2, b2, 1);
```

**Plotting:** figure for  $\gamma = 1$

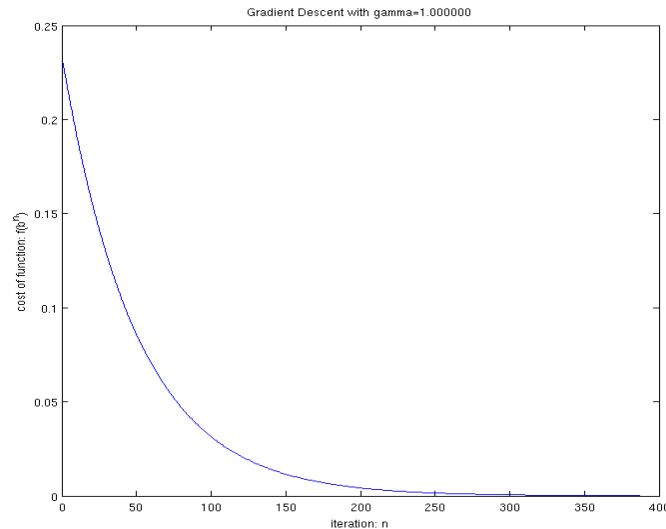


Figure 4: Plotting figure for gradient descent with  $\gamma = 1$  on the second matrix

**Explanation:** Through the smooth plotted curve, we guess that the gradient descent method got linear convergence when  $\gamma = 1$  on  $X_2$ . Hence, we trace convergence rate  $\text{conv\_rate} = f(x^k)/f(x^{k-1})$  as follows:

```
Iter: 2, Cost: 2.254428e-01, Conv_Rate: 0.980100
Iter: 3, Cost: 2.209565e-01, Conv_Rate: 0.980100
Iter: 4, Cost: 2.165594e-01, Conv_Rate: 0.980100
Iter: 5, Cost: 2.122499e-01, Conv_Rate: 0.980100
Iter: 6, Cost: 2.080261e-01, Conv_Rate: 0.980100
Iter: 7, Cost: 2.038864e-01, Conv_Rate: 0.980100
...
...
Iter: 381, Cost: 1.107934e-04, Conv_Rate: 0.980100
Iter: 382, Cost: 1.085886e-04, Conv_Rate: 0.980100
Iter: 383, Cost: 1.064277e-04, Conv_Rate: 0.980100
Iter: 384, Cost: 1.043098e-04, Conv_Rate: 0.980100
Iter: 385, Cost: 1.022340e-04, Conv_Rate: 0.980100
Iter: 386, Cost: 1.001996e-04, Conv_Rate: 0.980100
Iter: 387, Cost: 9.820558e-05, Conv_Rate: 0.980100
Converged to zeros!
```

In terms of above dumps and the fact that  $f(x^*) = 0$ , we can conclude that when  $\gamma = 1$

$$f(x^{k+1}) - f(x^*) = 0.9801 \cdot (f(x^k) - f(x^*))$$

which supports our previous guess that

Gradient Descent with  $\gamma = 1$  on second matrix leads to **linear convergence**.

## Part II

# Written Problems

### 1 Othorognal Subspace

(a) Show that if  $U$  is a subspace, then so is  $U^\perp$

*Proof.* Since  $U$  is a subspace, then we have  $U$  satisfying all three properties shown below:

- $\mathbf{0} \in U$
- $\forall \mathbf{u}_1, \mathbf{u}_2 \in U, \mathbf{u}_1 + \mathbf{u}_2 \in U$
- $\forall \mathbf{u} \in U, \alpha \in \mathbb{R}, \alpha \mathbf{u} \in U$

Now we show that  $U^\perp$  is also a subspace by indicating  $U^\perp$  satisfies all three properties as above  $U$  do.

- Since  $\forall \mathbf{u} \in U, \langle \mathbf{0}, \mathbf{u} \rangle = 0$  and  $\mathbf{0} \in V$  ( $\mathbf{0} \in U \subseteq V$ ), then it turned out that  $\mathbf{0} \in U^\perp$ .
- Let  $\mathbf{u}$  be arbitrary vector s.t.  $\mathbf{u} \in U$ , and  $\mathbf{x}_1, \mathbf{x}_2$  to be distinct vector s.t.  $\mathbf{x}_1 \in U^\perp$  and  $\mathbf{x}_2 \in U^\perp$ . By definition of  $U^\perp$ , we have  $\langle \mathbf{x}_1, \mathbf{u} \rangle = 0$  and  $\langle \mathbf{x}_2, \mathbf{u} \rangle = 0$ . Then it is obvious that  $\langle \mathbf{x}_1 + \mathbf{x}_2, \mathbf{u} \rangle = 0$ . That is  $\mathbf{x}_1 + \mathbf{x}_2 \in U^\perp$ . Therefore,  $\forall \mathbf{x}_1, \mathbf{x}_2 \in U^\perp, \mathbf{x}_1 + \mathbf{x}_2 \in U^\perp$  was proved.
- Let  $\mathbf{x}$  be arbitrary vector s.t.  $\mathbf{x} \in U^\perp$ ,  $\mathbf{u}$  be arbitrary vector s.t.  $\mathbf{u} \in U$  and arbitrary  $\alpha \in \mathbb{R}$ . By definition of  $U^\perp$ , we have  $\langle \mathbf{x}_1, \mathbf{u} \rangle = 0$ . Since inner product is linear operator, it is obvious that  $\langle \alpha \mathbf{x}_1, \mathbf{u} \rangle = 0$ . That is  $\alpha \mathbf{x}_1 \in U^\perp$ . Therefore,  $\forall \mathbf{x} \in U^\perp, \alpha \in \mathbb{R}, \alpha \mathbf{x} \in U^\perp$  was proved.

Since  $U^\perp$  contains  $\mathbf{0}$ , and is closed under addition and scalar multiplication, it turned out that  $U^\perp$  is a subspace. Therefore, the statement that if  $U$  is a subspace, then so is  $U^\perp$  was proved.  $\square$

(b) Show that  $(U^\perp)^\perp = U$

*Proof by contradiction.* Assume that  $(U^\perp)^\perp \neq U$  and then show the contradiction. By definition of  $U^\perp$ , we have  $U^\perp = \{\forall \mathbf{v} \in V, \langle \mathbf{v}, \mathbf{u} \rangle = 0, \forall \mathbf{u} \in U\}$  and  $(U^\perp)^\perp = \{\forall \mathbf{x} \in V, \langle \mathbf{x}, \mathbf{v} \rangle = 0, \forall \mathbf{v} \in U^\perp\}$ . Since  $(U^\perp)^\perp \neq U$ , then we can say that  $\exists \mathbf{x} \notin U, \langle \mathbf{x}, \mathbf{v} \rangle = 0, \forall \mathbf{v} \in U^\perp$ . That is to say,  $\exists \mathbf{x} \in V$  but  $\notin U, \langle \mathbf{x}, \mathbf{v} \rangle = 0, \forall \mathbf{v} \in U^\perp$  s.t.  $\langle \mathbf{v}, \mathbf{u} \rangle = 0, \forall \mathbf{u} \in U$ . However, such  $\mathbf{x}$  does not exist. Hence, we reject initial assumption and conclude that  $(U^\perp)^\perp = U$ .  $\square$

(c) Show that if  $U, W \subseteq V$  are subspaces of  $V$ , then  $U \subseteq W \Leftrightarrow U^\perp \supseteq W^\perp$

*Proof of  $U \subseteq W \Rightarrow U^\perp \supseteq W^\perp$ .*  $U^\perp = \{\forall \mathbf{v} \in V, \langle \mathbf{v}, \mathbf{u} \rangle = 0, \forall \mathbf{u} \in U\}$  and  $W^\perp = \{\forall \mathbf{v} \in V, \langle \mathbf{v}, \mathbf{w} \rangle = 0, \forall \mathbf{w} \in W\}$   $\square$

*Proof of  $U^\perp \supseteq W^\perp \Rightarrow U \subseteq W$ .*  $\square$

(d) Show that  $X^\perp$  makes sense,  $X^\perp$  is a subspace and  $(X^\perp)^\perp \supseteq X$

(e) Show that any  $v \in V$  can be written uniquely as  $v = u + u^\perp$

## 2 Boyd and Vandenberghe, Ex. 2.10

- (a) Show that if  $A \in \mathbb{S}_+^n$  then the set  $\mathbf{C}$  is convex
- (b) Show that  $C_1$  is convex if there exists  $\lambda \in \mathbb{R}$  such that  $(A + \lambda g g^T) \in \mathbb{S}_+^n$



### 3 Boyd and Vandenberghe, Ex. 2.21

## 7 Form a Half-Space

**8   Exists  $C$  such that  $CA = B$**

## A Codes Printout

### (a) Gradient Descent Routine

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Problem set 1: Standard Gradient Descent with fixed step size
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Usage:
%   [b, iter, all_costs] = gd (X, b_init, gamma)
% Parameters:
%   X: matrix in quadratic optimization
%   b_init: starting vector of variable b
%   gamma: fixed step size
% Note that stopping criteria is set by absolute eps = 10e-5.

function [b, iter, all_costs] = gd (X, b_init, gamma)
eps = 10e-5;
b = b_init;
last_cost = 0.5 * b' * X * b;
iter = 1;
all_costs = [];
while true,
    %% compute essential numerics and do gradient descent
    gradient = X * b;
    b = b - gamma * gradient;
    cost = 0.5 * b' * X * b;
    rate = (cost / last_cost);
    all_costs = [all_costs cost];
    %% output numeric information of this iteration
    disp(sprintf('Iter: %d, Cost: %e, Conv.Rate: %f', iter, cost, rate));
    %% quadratic optimization converges to zero
    if cost < eps,
        disp('Converged to zeros!')
        break
    end
    %% quadratic optimization diverges
    if cost >= last_cost && iter > 10,
        disp('Problem diverges!')
        break
    end
    %% prepare for next iteration
    last_cost = cost;
    iter = iter + 1;
end

%% uncomment following code for plotting individual gradient descent run
%% plot f(b^(n)) with regard to n
%% plot (1:iter, all_costs)
%% title (sprintf ('Gradient Descent with gamma=%f', gamma))
%% xlabel ('iteration: n')
%% ylabel ('cost of function: f(b^n)')

end

```

**(b) Running Script**

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Running scripts for applying gradient descent
%% on three given dataset
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% for X1, b1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gamma1_one = 0.5; gamma2_two = 3;
[b1_opt_one, iter1_one, costs1_one] = Gradient_Descent(X1, b1, gamma1_one);
[b1_opt_two, iter1_two, costs1_two] = Gradient_Descent(X1, b1, gamma2_two);
subplot (1, 2, 1)
plot (1:iter1_one, costs1_one)
axis ([-0.1*iter1_one 1.1*iter1_one -0.05*max(costs1_one) max(costs1_one)])
title (sprintf ('Gradient Descent with gamma=%f', gamma1_one))
xlabel ('iteration: n')
ylabel ('cost of function: f(b^n)')
subplot (1, 2, 2)
plot (1:iter1_two, costs1_two)
axis ([-0.1*iter1_two 1.1*iter1_two -0.05*max(costs1_two) max(costs1_two)])
title (sprintf ('Gradient Descent with gamma=%f', gamma2_two))
xlabel ('iteration: n')
ylabel ('cost of function: f(b^n)')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% for X2, b2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gamma2_one = 1.5; gamma2_two = 3;
[b2_opt_one, iter2_one, costs2_one] = Gradient_Descent(X2, b2, gamma2_one);
[b2_opt_two, iter2_two, costs2_two] = Gradient_Descent(X2, b2, gamma2_two);
figure()
subplot (1, 2, 1)
plot (1:iter2_one, costs2_one)
axis ([-0.1*iter2_one 1.1*iter2_one -0.05*max(costs2_one) max(costs2_one)])
title (sprintf ('Gradient Descent with gamma=%f', gamma2_one))
xlabel ('iteration: n')
ylabel ('cost of function: f(b^n)')
subplot (1, 2, 2)
plot (1:iter2_two, costs2_two)
axis ([-0.1*iter2_two 1.1*iter2_two -0.05*max(costs2_two) max(costs2_two)])
title (sprintf ('Gradient Descent with gamma=%f', gamma2_two))
xlabel ('iteration: n')
ylabel ('cost of function: f(b^n)')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% for X3, b3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gamma3_one = 0.005; gamma3_two = 0.05;
[b3_opt_one, iter3_one, costs3_one] = Gradient_Descent(X3, b3, gamma3_one);
[b3_opt_two, iter3_two, costs3_two] = Gradient_Descent(X3, b3, gamma3_two);
figure()
subplot (1, 2, 1)
plot (1:iter3_one, costs3_one)
axis ([-0.1*iter3_one 1.1*iter3_one -0.05*max(costs3_one) max(costs3_one)])
title (sprintf ('Gradient Descent with gamma=%f', gamma3_one))
xlabel ('iteration: n')
ylabel ('cost of function: f(b^n)')
subplot (1, 2, 2)
plot (1:iter3_two, costs3_two)
axis ([-0.1*iter3_two 1.1*iter3_two -0.05*max(costs3_two) max(costs3_two)])
title (sprintf ('Gradient Descent with gamma=%f', gamma3_two))
xlabel ('iteration: n')
ylabel ('cost of function: f(b^n)')

```