الأسبوع الثاني

Linear Regression with Multiple Variables

- التعامل مع اكثر من بعد :
  - تحدثنا سابقا , عن التعامل مع متغير واحد (قيمة لاكس و نجيب منها قيمة واي) الان نتعامل مع اكثر من متغير
  - أكثر من متغير معناها ان البيانات الداخلة لها اكثر معلومة لكل صف , فبدلا من ادخال مساحة البيت لمعرفة سعره (اكس واحدة) , نقوم بادخال مساحة البيت و عدد غرفه , وعمره, و موقعه , وحالته , ولونه , لتحديد سعره , وهذه الأشياء تسمي features

Multiple features (variables).

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$m = 47$

Notation:

n = number of features $n = 4$

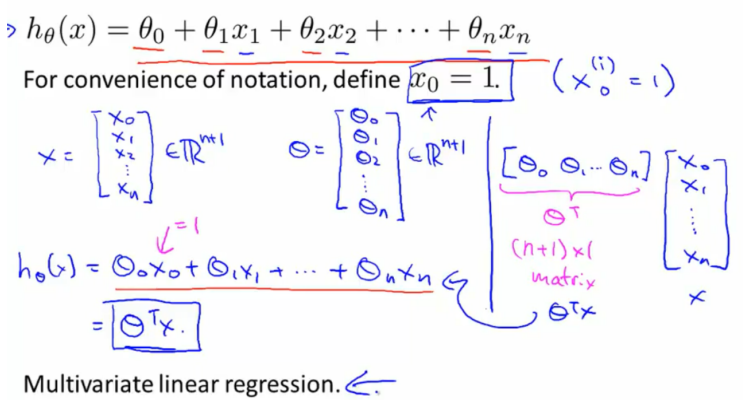
$x^{(i)}$ = input (features) of i^{th} training example.

$x_j^{(i)}$ = value of feature j in i^{th} training example.

  - فنري ان سعر البيت (Label Y) يتاثر بعدد من العوامل (Features Xs)
  - عدد الاكسات نسميه n, بينما عدد الصفوف لازال m
  - عشان منتلغبطش , هنعمل التسمية ديه

$$x_j^{(i)} = \text{value of feature } j \text{ in the } i^{th} \text{ training example}$$

  - الرقم اللي فوق يكون رقم الصف (انهي ريكورد فيهم m) و الرقم اللي تحت هيكون رقم العمود (انهي معلومة فيهم n)
  - وقتها الفنكشن , هتكون متعددة الحدود زي كدة , وهنعمل ماتركس للاكسات , وواحدة للثيتات , ونضربهم في بعض بعد ما نعمل ترانزبوس لوحدة فيهم
  - ليه بنعمل ترانزبوس ؟ لان الثيتا و الاكس اصلا هما فيكتورات (عمود واحد في كذا صف) , فلازم اعمل ترانزبوس لواحد فيهم و اضربه في التاني , عشان تكون المصفوفة الاولي صف واحد في 5 عواميد مثلا , والتانية زي ما هي 5 صفوف في عمود واحد , يتضربو يبقو رقم واحد بس



- وخد بالك الصيغة القديمة اللي كانت للـ J هيكون فيها شوية تعديل , عشان مبقاش عامل واحد , نفس المعادلة , لكن دلوقتي H بقت فيها ثيتات كتيرة
- لما اعمل تفاضل , هتظل اتش زي هي , وهيموت كل الثيتات التانية عدا الثيتا اللي باعمل تفاضل علي اساسها اللي هتتبقي في الاخر

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

...

}

- عشان كدة ممكن اعمل صيغة عامة زي ديه

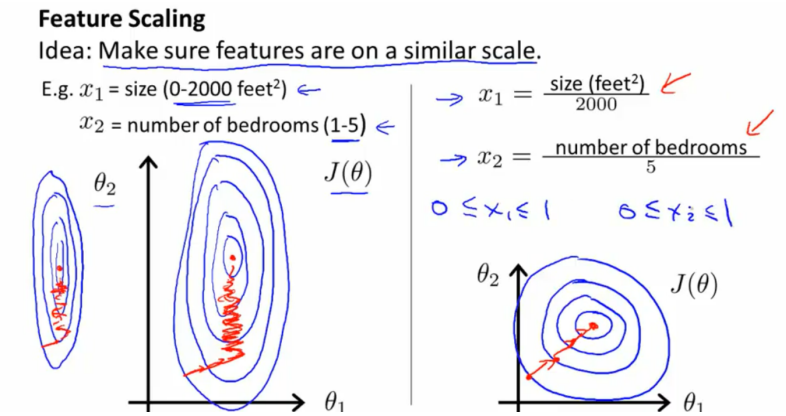
repeat until convergence: {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \qquad \text{for j} := 0...n$$

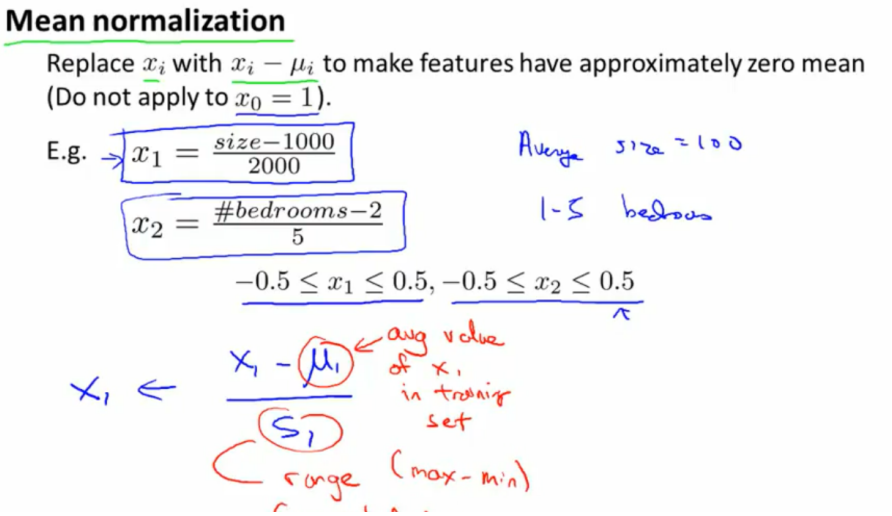
}

- لاحظ ان ثيتا 0 مكانش فيها اكس صفر , بس مفيش مشاكل من وضعه لان اساسا اكس صفر = 1

- اعادة التكبير Rescaling
  - لو كانت قيم اكس 1 و اكس اتنين بتدريجات مختلفة , يعني الاول ينتهي عند 5 و التاني 2000 مثلا , هنلاقي الكونتورز لرسم الـ J طويلة و رفيعة او العكس , وده هيخلي عملية الـ gradient descent طويلة جدا
  - فالحل اننا نخلي سكيل جميع الاكسات علي 1 بس , فنقسم اي قيمة لاكس 1 علي القيمة الاقصي لاكس واحد

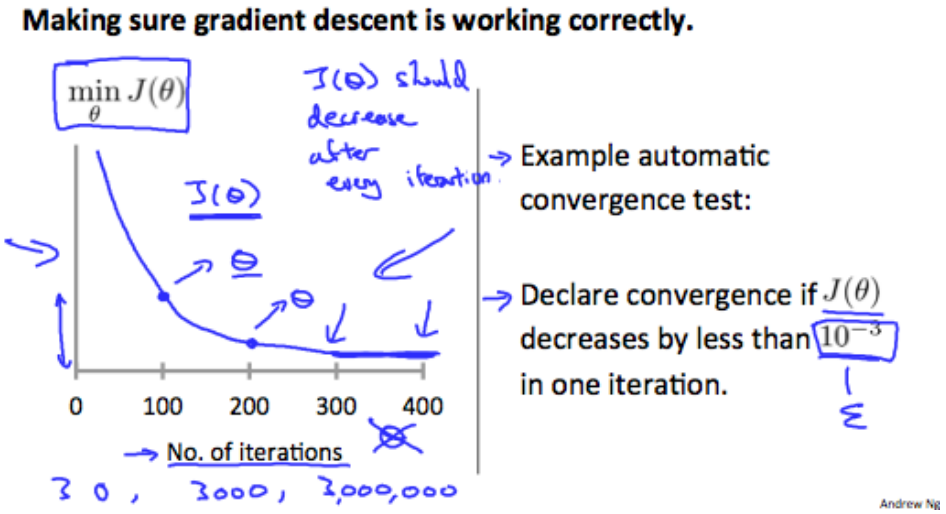


- كدة هيتحول الرسم لعادي , وهتكون قيم اكس كلها بين 0 وواحد والعملية هتكون اسرع
- كمان لمزيد من الدقة و السرعة فيما بعد , ممكن نطرح كل قيم اكس , ناقص المتوسط (هيتم اعطاؤه او نجيبه احنا علي الرينج (الرقم الاكبر ناقص الرقم الاصغر)

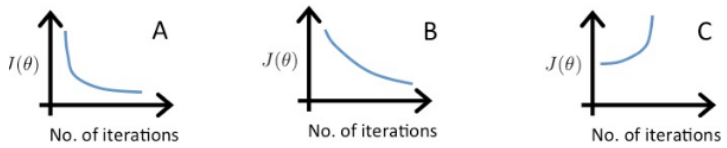


- وده هيخلي غالبا ارقام اكسات كلها بتلعب من 1 لسالب 1, وبشكل عام , مفيش مشكلة لو كان الرقم بين 3 و سالب 3

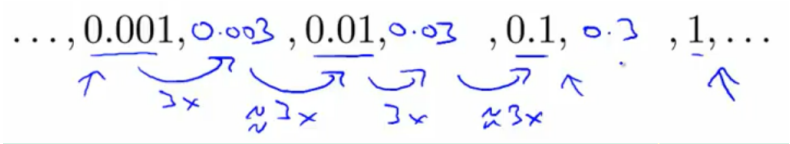
- احاول كام مرة ؟ ؟
  - من الواضح ان كل ما بنحاول اكتر , قيمة J بتقل و ديه حاجة كويسة , بس ياتري هنحاول كام مرة ؟ ؟
  - الرسمة هنا واضح فيها ان كل ما بنزود عدد المحاولات , كل ما قيمة J هتقل اكتر , بس بعد فترة معينة السلوب بيقرب لصفر , و بيكون فيه عدد ضخم جدا من المحاولات مع فرق بسيط , و هنا لازم نوقف , عشان هيكون ضياع وقت علي الفاضي
  - ممكن نوقف بعد 5 او 50 او 5 مليون محاولة , محدش هيقدر يحدد الرقم كام , كل حالة بحالتها



- تحديد قيمة الفا
  - لو زادت قيمة الفا هجري بسرعة ,بس ممكن اقع في مشكلة اني ازود قيمة الـ J , ولو مشيت ببطئ , هيكون دقيق بس بطئ جدا , فلازم اختار قيم مظبوطة



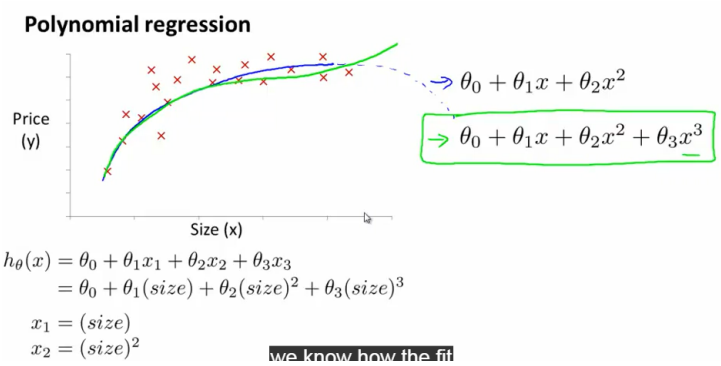
  - علي اليمين قيمة كبيرة , في النص قيمة صغيرة , علي الشمال قيمة معقولة
  - و ممكن نختار القيم كالتالي :



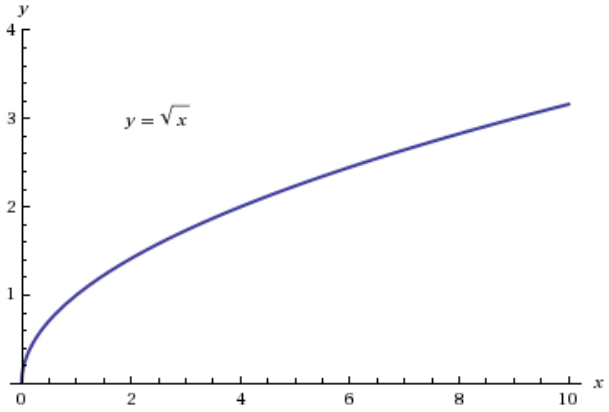
  - هات 0.001 و جرب , لو الخطوات صغيرة , اضرب في 3 , يعني 0.003 , لسة صغيرة , اضرب في 3 يعني 0.01 و هكذا , اول ما الاقي القيمة كبرت ارجع خطوة ورا

- التعديل في عدد البيانات
  - لو عندنا بيت , وسعره يقيم علي اساس طوله و عرضه , فممكن نعمل حاجتين
    - بدل ما يكون الاعتماد علي عنصرين X2 , X1 فهتكون معادلة فيها 3 ثيتا و تغلبنا , نجيب المساحة مكانها باعتبارها تشمل الطول و العرض , وتكون معالة خطية X1 بس
    - اننا نجيب المساحة و تكون بيان اضافي للطول و العرض فيكون فيه X3 , X2, X1

- اختيار نوع الدالة



  - لاحظ ان النقط الحمرا و هي البيانات , لو همثلها بمعادلة خطية (بناء علي المتغير X) فممكن متدينيش الخط المناسب
  - فممكن افكر في معادلة تربيعية (بناء علي X , X تربيع) ,بس المعادلة التربيعية بتؤول انها تنزل تحت و ده غلط عمليا (سعر البيت مش هيقل بزيادة المساحة) فالافضل اخليها معادلة تكعيبية اللي بتطلع لفوق (اس , اكس تربيع , اكس تكعيب) و ده صح منطقيا
  - الكلام اللي فات ده كان علي دالة سعر البيت , بس وارد المعادلة التربيعية تكون مناسبة مثلا لعدد العمال اللي هجيبهم المصنع و المكسب (لو زاد عدد العمل عن المطلوب هيعمل خساير و ده منطقي)
  - ممكن استعين بالمعادلة الجذرية (روت اكس جنب الاكس) , وديه بتخلي الرسم (flattered) قدام , وده وارد يكون مناسب لنوع معين من المشاكل



- طريقة الـ Normal Equation
  - و هي عن طريق الاعتماد علي تفاضل الـ J و مساوتها بالصفر لايجاد قيمة الثيتا المطلوبة

Examples: $m = 4$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \quad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

m x (n+1) — m-dimensional vector

- و اذا كان لدينا جدول مثل هذا لاكثر من متغير , فبعد مفاضلتها و مساوتها بالصفر ستكون الثيتا كالتالي

$$\theta = (X^T X)^{-1} X^T y$$

- و متنساش ان دايما ماتركس اكسات فيها اول عمود وحايد بس (عشان يتضرب في ثيتا صفر و يعمل نفس قيمتها) , بعدها قيم اكسات , بينما ماتركس واي هي فيكتور لقيم واي تحت بعض
- أيهما أفضل : الجرادينت ولا النورمال اكواشن :
  - النورمال ميزتها ان مش محتاج تحسب قيمة الفا , و مش هتعمل خطوات كتير , هي خطوة واحدة
  - بس عيبها انها بتكون صعبة و بطيئة جدا مع عدد كبير للخواص n لان الماتركس هتكون مخيفة خاصة لعمل الانفرس , فلو عدد الـ N يقول عن 10 الاف خليك في النورمال , زادت روح للجرادينت
  - كمان فيه خوارزميات (زي linear regression) مش هينفع تشتغل الا بالجرادينت , و خوارزميات تانية ممكن بالنورمال , فلازم تكون عارف الاتنين و تشوف مين مناسب لايه

- أحيانا بتحصل مشكلة في نوع النورمال , ان ماتركس اكس ترانسبوز في اكس تكون singular و معناها ان مش هينفع يتعمل لها انفرس , وده هيلغبط الدنيا
- غالبا بيكون سبب انها انفرس حاجة من اتنين
  - اما يكون فيه عمودين معتمدين علي بعض , يعني فيه مثلا مساحة البيت بالمتر المربع , ومساحة البيت بالقدم المربع , وده معناه ان فيه عمود كامل يساوي عمود تاني مضروب في فاكتور , وده هيؤدي ان قيمة الماتركس كلها تساوي صفر , فالانفرس هيختفي
  - ان عدد الـ m (عدد الصفوف) اقل من عدد الـ n (العواميد او المعلومات عن كل بيت) خاصة لو الفرق كبير , فا اما تمسح شوية عواميد مش مهمة , او تزود بيانات و صفوف , او تشوف نوع تاني

التعلم علي كود برنامج Octave من الملف الخاص به

- مثال عملي : هنجيب دالة الكوست باستخدام اوكتيف

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

- دالة الكوست عبارة عن : واحد علي ضعف عدد السامبل , مضروبة في مجموع مربعات فروق قيمة الاتش من قيمة الواي
- قيمة الاتش عبارة حاصل ضرب الاكسات في الثيتا
- الأول هنكتب الكود في ملف خارجي بامتداد m وهو :

```
function J = costfunctionJ( x,y,theta)
m = size(x,1);
predictions = x*theta;
sgrerrors = (predictions -y).^2;
J = (1/(2*m))* sum(sgrerrors );
```

  - اول سطر تعريف للفنكشن , والمدخلات و المخرج
  - تاني حاجة باعرف الفنكشن , ان الام هي عدد صفوف الماتركس اكس اللي هدخلها
  - بعدها باعمل ماتركس جديدة اسمها بريدكشنز (اللي هي كانها قيم H), وهي حاصل ضرب ماتركس اكس في ماتركس ثيتا
  - لاحظ ان قيم اكس , انا اضفت عليها عمود علي الشمال (قيمة X0) اللي هو وحايد , عشان لما يتضرب في ثيتا 0 يكون نفس القيمة , ولما قيم اكس تتضرف في ثيتا واحد يدونا الرقم المطلوب
  - ضرب المصفوفتين هيعملنا مصفوفة جديدة اللي هي البريديكشن
  - دلوقتي نطرح مصفوفة البريديكشن من مصفوفة الواي عشان اجيب الفرق
  - بعدها اعمل سكوير لقيمهم بس (مش ماتركس سكوير) عشان كدة عملت بوينت قبل ما اعمل سكوير , عشان اقوله عايز سكوير لقيمهم
  - اخر حاجة اني اضرب 1 علي ضعف الـ m , مضروبة في مجموع كل قيم الماتركس الاخيرة

- دلوقتي ندخل البيانات
  - >> x = [1 1;1 2;1 3] مصفوفة اكس مضاف اليها وحايد علي الشمال و هتبقي كدة

```
1   1
1   2
1   3
```

  - >> y = [1;2;3] قيم واي , وهتبقي كدة

```
1
2
```

- احنا عارفين القيم , بس بفرض اننا مش عارفينها , وديه القيم اللي ممكن قدام نغيرها عشان [0;1] = theta >> نجيب افضل قيم ليها
- اكتب الفنكشن داخل فيها التلات حاجات دول , يجيبلي جي (costfunctionJ(x,y,theta = j >>
  - ولو غيرنا اي قيمة في اكسات او وايات او ثيتات هتلاقي الناتج اختلف

تتبع الاختبار الاول

- هي مجموعة من الملفات المرتبطة ببعض , ويقودها الملف الرئيسي ex1 والذي يذهب الي باقي الملفات و الفنكشنز

Ex1()	
%% Machine Learning Online Class - Exercise 1: Linear Regression % Instructions % ------------ % This file contains code that helps you get started on the % linear exercise. You will need to complete the following functions % in this exericse: % % warmUpExercise.m % plotData.m % gradientDescent.m % computeCost.m % gradientDescentMulti.m % computeCostMulti.m % featureNormalize.m % normalEqn.m % For this exercise, you will not need to change any code in this file, % or any other files other than those mentioned above. % % x refers to the population size in 10,000s % y refers to the profit in $10,000s % %% Initialization	كل هذه مقدمة يشرح فيها المطلوب
clear ; close all; clc	لتصفير البيانات كلها
%% ==================== Part 1: Basic Function ====================	لطباعة كلمة كذا و كذا

% Complete warmUpExercise.m
 	
fprintf('Running warmUpExercise ... \n');
fprintf('5x5 Identity Matrix: \n');
warmUpExercise()
	للذهاب للفنكشن وارم اب

warmUpExercise()	
function A = warmUpExercise() A = eye(5); end	اسم الدالة , وتكوين دالة ادينتيتي 5 في 5
Ex1()	
fprintf('Program paused. Press enter to continue.\n'); pause;	لكتابة الجملة , ثم التوقف
fprintf('Plotting Data ...\n')	كتابة الجملة
data = load('ex1data1.txt');	تحميل الملف اللي فيه الداتا
X = data(:, 1); y = data(:, 2);	الاكس فيكتور يساوي جميع الارقام اليسري , والواي تساوي الارقام اليمني
m = length(y); % number of training examples	الإم رقم يساوي عدد صفوف الواي او الاكس
plotData(X, y);	نذهب لفنكشن الرسم
PlotData()	
function plotData(X, y)	اسم الدالة و متغيراتها
plot (X, y, 'rx', 'MarkerSize', 10);	لرسم الدالة , ورقم 10 يعتبر حجم النقطة , و ار اكس و الماركر متغيرات لتحديد الرسم
ylabel('Profit in $10,000s'); xlabel('Population of City in 10,000s');	لكتابة العناوين علي محوري اكس وواي
figure; % open a new figure window	لاظهار الرسم
Ex1()	
X = [ones(m, 1), data(:,1)]; % Add a column of ones to x	لاضافة عمود بعدد ام علي يسار الاكس بحيث يكون مصفوفة ام في 2 مش في 1
theta = zeros(2, 1); % initialize fitting parameters	عمل ثيتا مصفوفة اصفار 2 في 1
iterations = 1500; alpha = 0.01;	عدد المحاولات , وقيمة الفا

fprintf('\nTesting the cost function ...\n')	اظهار كلام
J = computeCost(X, y, theta);	حساب الجي بالدالة ديه
ComputeCost()	
function J = computeCost(X, y, theta)	تعريف الدالة
m = length(y); J = 0;	تحديد قيمة ام , وقيمة اولية للجي
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$ Parameters: θ_0, θ_1 Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$ Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\, J(\theta_0, \theta_1)$	هنا المعادلة المطلوبة
f = (X * theta);	هنعمل مصفوفة لضرب قيم اكس في ثيتا
g = f-y;	هنعمل مصفوفة تانية لطرق قيمة المصفوفة اللي عملناها من واي
h = g.^2;	مصفوفة جديدة لتكوين مربعات قيم المصفوفة اللي فاتت
o = sum(h);	هنجمع قيم المصفوفة علي بعض
J = o / (2*m)	اخيرا , هنقسم قيمة المجموع علي 2 في ام
Ex1()	
fprintf('With theta = [0 ; 0]\nCost computed = %f\n', J); fprintf('Expected cost value (approx) 32.07\n');	رغي
J = computeCost(X, y, [-1 ; 2]);	حساب الجي مرة اخري بنفس الدالة

fprintf('\nWith theta = [-1 ; 2]\nCost computed = %f\n', J);
fprintf('Expected cost value (approx) 54.24\n');
fprintf('Program paused. Press enter to continue.\n');
Pause;
fprintf('\nRunning Gradient Descent ...\n')
	رغي

theta = gradientDescent(X, y, theta, alpha, iterations);	تعريف الفنكشن الي هنروحلها
gradientDescent()	

function [theta, J_history] = gradientDescent(X, y, theta, alpha, num_iters)	اسم الدالة , و هيطلع منها معلومتين , مصفوفة ثيتا النهائية (2 في 1) , وجي هيستوري , وديه ماتريكس عمود واحد فيها كل قيم جي اللي طلعت
m = length(y); J_history = zeros(num_iters, 1);	تحديد ام , و تكوين جي هيستوري , بحيث يكون عدد صفوفها هو عدد المحاولات اللي تم تحديده من شوية 1500
for iter = 1:num_iters	نبدا الفور , ولعدد 1500 مرة
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$ $$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i)x_i)$$	هنا الدالة اللي هنمشي عليها , الثيتا هي الثيتا الاصلية , د الف علي ام , مضروبة في مجموع قيم اتش ناقص واي و ثيتا واحد زيها بس المجموع مضروب في اكس 1 الاول
th1 = theta(1,1); th2 = theta(2,1);	تحديد متغيرين ياخدو قيم ثيتا من المصفوفة
f = (X * theta);	مصفوفة جديدة ضرب اكس في ثيتا (1*97)

g1 = f-y;
o1 = sum(g1);
l1 = (alpha * o1) / m;
	طرحها من واي (1*97)
	مجموعها (رقم)
	اضرب في الفا علي إم

s2 = X(:,2);
g2 = (f-y).*s2;
o2 = sum(g2);
l2 = (alpha * o2) / m;
	مصفوفة تشيل قيم اكس علي اليمين من غير وحايد (1*97)
	اطرح و اضرب القيم فيها
	اجيب محموعها و اقسمه علي ام في الفا

th1 = th1 - l1;
th2 = th2 - l2;
theta = [th1 ; th2];
	اطرح كل ده من الثيتات , واعمل ابديت لمصفوفة الثيتا

J_history(iter) = computeCost(X, y, theta); end	احسب قيمة ثيتا بالمعادلة القديمة , واحطها في جي هيستوري , واقفل الفور
Ex1()	

fprintf('Theta found by gradient descent:\n');
fprintf('%f\n', theta);
fprintf('Expected theta values (approx)\n');
fprintf(' -3.6303\n 1.1664\n\n');
	عرض قيمة الثيتا

hold on; % keep previous plot visible	سيب الرسمة القديمة متقفلهاش

plot(X(:,2), X*theta, '-') legend('Training data', 'Linear regression') hold off % don't overlay any more plots on this figure	لرسم الخط البيست فيت لتحديد القيم المكتوبة خلاص اقفل الرسمة

predict1 = [1, 3.5] *theta;
fprintf('For population = 35,000, we predict a profit of %f\n',...
predict1*10000);
predict2 = [1, 7] * theta;
fprintf('For population = 70,000, we predict a profit of %f\n',...
predict2*10000);
	هنطبق الثيتا , عن طريق ضرب رقم 1 , و 3.5 في الثيتتين , و نشوف قيمة الواي كام , ونطبق كمان مر , ونعرض النتايج

fprintf('Program paused. Press enter to continue.\n'); pause; fprintf('Visualizing J(theta_0, theta_1) ...\n')	رغي و التحضير لعرض الرسم المصور لثيتا
theta0_vals = linspace(-10, 10, 100); theta1_vals = linspace(-1, 4, 100);	هنعمل قيم كتيرة للثيتا 0 و ثيتا 1 , عن طريق فنكشن بتعمل فيكتور , وهي عدد 100 رقم (الرقم الايمن) يبدا من سالب 10 (الايسر) و ينتهي عند 10 (الاوسط) و زيها اللي تحت
J_vals = zeros(length(theta0_vals), length(theta1_vals));	هنعمل مصفوفة اصفار , هيكون بعديها طولي الرقمين اللي فوق , يعني 100*100
for i = 1:length(theta0_vals) for j = 1:length(theta1_vals) t = [theta0_vals(i); theta1_vals(j)]; J_vals(i,j) = computeCost(X, y, t); end end	هنملي المصفوفة الكبيرة 100*100 , بقيمة كوست كل مرة , بحيث كل قيمة نعوض فيها بمقدار ثيتا صفر , وثيتا 1 و الاكس و الواي الثابتين , ونملي بيها المصفوفة الكبيرة
J_vals = J_vals';	هنعمل ترانزبوس للمصفوفة الكبيرة , لان الرسام مش هيعرف يتعامل معاها كدة
figure; surf(theta0_vals, theta1_vals, J_vals) xlabel('\theta_0'); ylabel('\theta_1');	الدالو سريف بترسم ثلاثي الابعاد , وهنحط قيم ثيتتين , ومع مكتبة الجي
contour(theta0_vals, theta1_vals, J_vals, logspace(-2, 3, 20)) xlabel('\theta_0'); ylabel('\theta_1'); hold on;	لرسم الدوائر البيضاوية
plot(theta(1), theta(2), 'rx', 'MarkerSize', 10, 'LineWidth', 2);	لتحديد موضع الثيتا النموذجية