COMP6065 – ARTIFICIAL INTELLIGENCE

Programming Exercise 1: Linear Regression

People
Innovation
Excellence

Master of Information
Technology – Master Track

Disusun oleh:

Gregorius Natanael E. – 2101629022

Jason – 2101700371

Nicholas Dominic – 2101628594

UNIVERSITAS BINA NUSANTARA

JAKARTA

2018

Daftar Isi

BAB I

PENDAHULUAN

A. Landasan Teori

Ketika mendapatkan satu *set* data, mungkin kita berpikir: bagaimana *kecenderungan* data-data tersebut di masa depan? Jawabannya adalah melalui *linear regression*.

Regresi adalah proses identifikasi relasi dan pengaruhnya terhadap nilai-nilai objek, yang bertujuan untuk menemukan suatu fungsi yang memodelkan data dengan meminimalkan selisih antara nilai prediksi dengan nilai yang sebenarnya.

Ada istilah lain yang sedikit berbeda: klasifikasi. Jika regresi bersifat kontinu, klasifikasi bersifat diskret. Contoh paling sederhana: jika kita berkata, "Besok harga emas diprediksi naik tajam," maka pernyataan ini merupakan klasifikasi; sedangkan jika kita berkata, "Besok harga emas diprediksi naik sebesar Rp 49,972.03," maka pernyataan ini merupakan regresi.

B. Pemahaman Secara Umum

Secara umum ada empat jenis regresi: regresi linear sederhana, regresi linear berganda, regresi non-linear sederhana, dan regresi non-linear berganda. Namun untuk membahas secara umum, kami akan menjabarkan mengenai turunan rumus regresi linear sederhana.

Regresi linear sederhana hanya melibatkan satu variabel bebas X. Sebuah fungsi linear akan membentuk garis lurus dengan persamaan $\boldsymbol{f}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + \boldsymbol{a}$. Model linear dapat diartikan sebagai penggunaan suatu fungsi lurus (untuk kasus regresi) atau suatu fungsi penilai (untuk kasus klasifikasi). Jadi, *learning* pada model linear adalah untuk menentukan dua parameter, yaitu **w** (vektor bobot / *weight*) dan $\boldsymbol{a}$ (*intercept* sebuah nilai tunggal skalar; dalam bidang *machine learning* biasa disebut sebagai *bias*) dari *training data set*.

Misalnya kita memiliki suatu *training data set* (x_1, y_1), … (x_n, y_n). Model linear dapat dinyatakan sebagai w, $a \in R$ sedemikian hingga meminimalkan kesalahan (*error*):

$$\mathrm{E}(\boldsymbol{w}, \boldsymbol{a}) = \sum_{i=1}^{n} (\boldsymbol{y_i} - (\boldsymbol{w x_i} + \boldsymbol{a}))^2$$

yang dikenal sebagai *ordinary least squares*. Masalah minimalisasi *error* ini dapat diselesaikan dengan membuat turunan parsial untuk masing-masing parameter w dan a.

$$\begin{aligned}\frac{\partial \mathrm{E}(w,a)}{\partial a} &= -2\sum_{i=1}^{n}(y_i - wx_i - a) = 0\\ y_i - wx_i - a &= 0\\ \boldsymbol{\bar{y} - w\bar{x}} &= \boldsymbol{a}\end{aligned}$$

di mana $\boldsymbol{a}$ adalah solusi untuk bias pada model linear yang kita harapkan, sedangkan $\bar{x}$ dan $\bar{y}$ adalah rata-rata nilai pada semua *x* dan *y* dari 1 sampai *n* yang dinyatakan sebagai $\boldsymbol{\bar{x}} = \frac{\sum_{i=1}^{n} x_i}{n}$ dan $\boldsymbol{\bar{y}} = \frac{\sum_{i=1}^{n} y_i}{n}$. Selanjutnya,

$$\begin{aligned}\frac{\partial \mathrm{E}(w,a)}{\partial w} &= -2\sum_{i=1}^{n}x_i(y_i - wx_i - a) = 0\\ x_i(y_i - wx_i - (\boldsymbol{\bar{y} - w\bar{x}})) &= 0\\ \frac{\sum_{i=1}^{n} \boldsymbol{x_i(y_i - \bar{y})}}{\sum_{i=1}^{n} \boldsymbol{x_i(x_i - \bar{x})}} &= \boldsymbol{w}\end{aligned}$$

Dengan demikian, telah diperoleh solusi untuk kedua parameter ***w*** dan ***a***. Persamaan di atas mengindikasikan bahwa, secara implisit, titik $(\bar{x}, \bar{y})$ **pasti** berada pada model regresi linear ***y* = *f*(x) = w · x + *a*** yang berupa **garis lurus** (lihat pada Bab II – Figure I, hlm. 5).

BAB II

LAPORAN KEGIATAN

1. Simple Octave/MATLAB Function

Pada bagian ini, kami membuat fungsi untuk membuat matriks identitas dengan menggunakan fungsi `eye()` dari Octave. Kami memahami bahwa fungsi ini akan mengembalikan sebuah matriks identitas dengan ukuran a × a di mana a adalah parameter yang berupa integer/bilangan bulat.

```
function A = warmUpExercise()
%WARMUPEXERCISE Example function in octave
%   A = WARMUPEXERCISE() is an example function that returns the 5x5 identity matrix

% ============= YOUR CODE HERE ==============
% Instructions: Return the 5x5 identity matrix
%               In octave, we return values by defining which variables
%               represent the return values (at the top of the file)
%               and then set them accordingly.

A = eye(5);




% ===========================================


end
```

```
Running warmUpExercise ...
5x5 Identity Matrix:
ans =

Diagonal Matrix

   1   0   0   0   0
   0   1   0   0   0
   0   0   1   0   0
   0   0   0   1   0
   0   0   0   0   1

Program paused. Press enter to continue.
```

2. Linear regression with one variable

a. Plotting the Data:

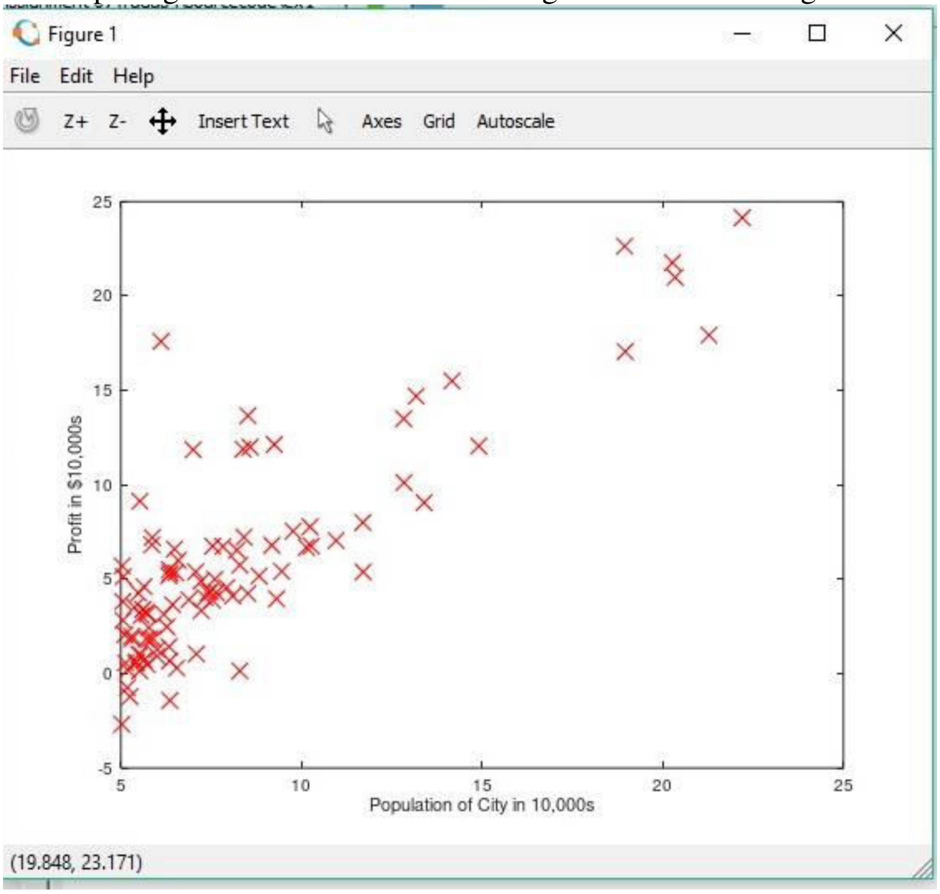
```
%% ======================= Part 2: Plotting =======================
fprintf('Plotting Data ...\n')
data = load('ex1data1.txt');
X = data(:, 1); y = data(:, 2);
m = length(y); % number of training examples

% Plot Data
% Note: You have to complete the code in plotData.m
plotData(X, y);

fprintf('Program paused. Press enter to continue.\n');
pause;
```

Pada bagian ini, kami membuat 2-D plot dengan menggunakan fungsi `plot()`. Pertama, data akan diinput dengan fungsi load dari ex1data1.txt. Data yang sudah diinput akan dimasukan ke dalam fungsi plotData.

warmUpExercise.m | ex1.m | plotData.m

```
function plotData(x, y)
%PLOTDATA Plots the data points x and y into a new figure
%   PLOTDATA(x,y) plots the data points and gives the figure axes labels of
%   population and profit.

figure; % open a new figure window

% ====================== YOUR CODE HERE ======================
% Instructions: Plot the training data into a figure using the
%               "figure" and "plot" commands. Set the axes labels using
%               the "xlabel" and "ylabel" commands. Assume the
%               population and revenue data have been passed in
%               as the x and y arguments of this function.
%
% Hint: You can use the 'rx' option with plot to have the markers
%       appear as red crosses. Furthermore, you can make the
%       markers larger by using plot(..., 'rx', 'MarkerSize', 10);

plot(x,y, 'rx', 'MarkerSize', 10);
ylabel('Profit in $10,000s');
xlabel('Population of City in 10,000s');


% ============================================================

end

```

Kami menambahkan sedikit fungsi pada plotData.m. Fungsi plot(x,y, 'rx', 'MarkerSize', 10); akan membuat 2-D plot dengan marker berupa Red X 'X' (rx) dengan ukuran 10. Label untuk sumbu X dan Y juga sudah ditulis pada gambar di atas dan akan menghasilkan hasil sebagai berikut.



b. Gradient Descent

Pada bagian ini, kami memasukkan fungsi *cost function* dan *gradient descent*, yang digunakan untuk meminimalisasi hasil *cost function*. Rumus *cost function* adalah:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

Kedua fungsi di atas dimasukkan ke dalam computeCost.m seperti gambar di bawah ini.

warmUpExercise.m | ex1.m | plotData.m | computeCost.m | gradientDescent.m

```
function J = computeCost(X, y, theta)
%COMPUTECOST Compute cost for linear regression
%   J = COMPUTECOST(X, y, theta) computes the cost of using theta as the
%   parameter for linear regression to fit the data points in X and y

% Initialize some useful values
m = length(y); % number of training examples

% You need to return the following variables correctly

% ====================== YOUR CODE HERE ======================
% Instructions: Compute the cost of a particular choice of theta
%               You should set J to the cost.
J = sum((X * theta - y) .^ 2) / (2*m);
% =========================================================================

end

```

Langkah berikutnya adalah mengimplementasikan *gradient descent* untuk memperkecil hasil J. Rumus *gradient descent* adalah sebagai berikut :

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

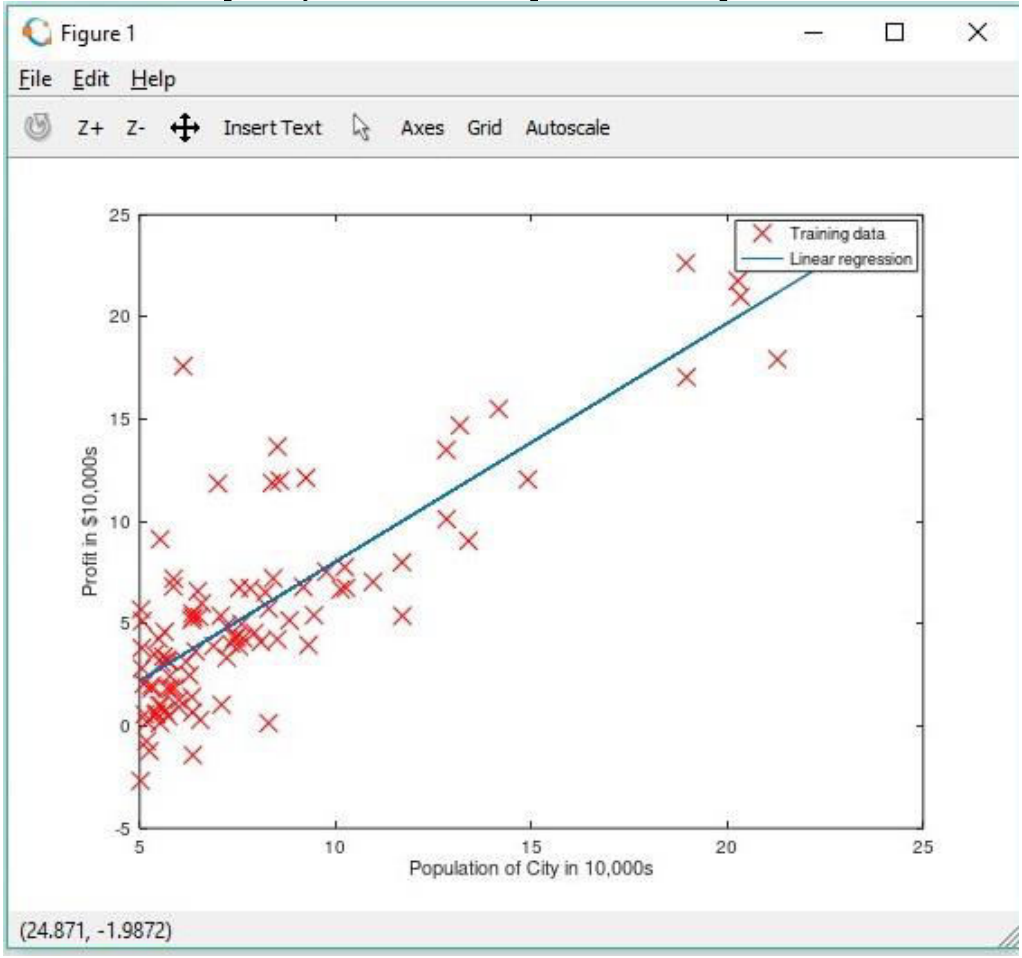
Setelah itu, nilai ϴ untuk setiap J harus diubah, atau dengan kata lain melakukan *compute cost* lagi untuk semua data yang ada. Maka hasil code adalah sebagai berikut:

warmUpExercise.m | ex1.m | plotData.m | computeCost.m | *gradientDescent.m

```
function [theta, J_history] = gradientDescent(X, y, theta, alpha, num_iters)
%GRADIENTDESCENT Performs gradient descent to learn theta
%   theta = GRADIENTDESCENT(X, y, theta, alpha, num_iters) updates theta by
%   taking num_iters gradient steps with learning rate alpha

% Initialize some useful values
m = length(y); % number of training examples
J_history = zeros(num_iters, 1);

for iter = 1:num_iters
    % ====================== YOUR CODE HERE ======================
    % Instructions: Perform a single gradient step on the parameter vector
    %               theta.
    %
    % Hint: While debugging, it can be useful to print out the values
    %       of the cost function (computeCost) and gradient here.
    %
    dJ = (X' * (X*theta-y))/m;
    theta = theta - alpha * dJ;
    % ============================================================
    % Save the cost J in every iteration
    J_history(iter) = computeCost(X, y, theta);
end

end
```

c. Debugging

Setelah itu, code pun dijalankan dan didapatkan hasil seperti berikut.



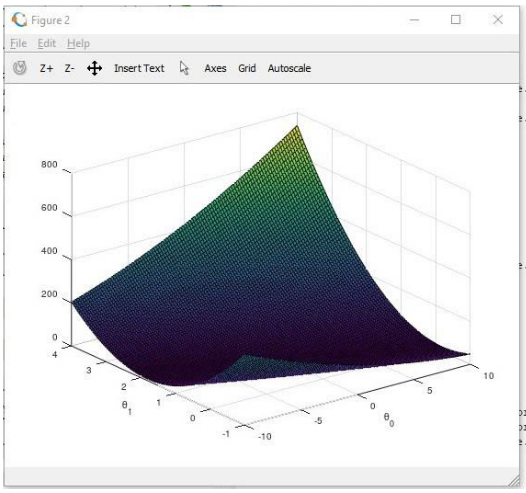
d. Visualizing

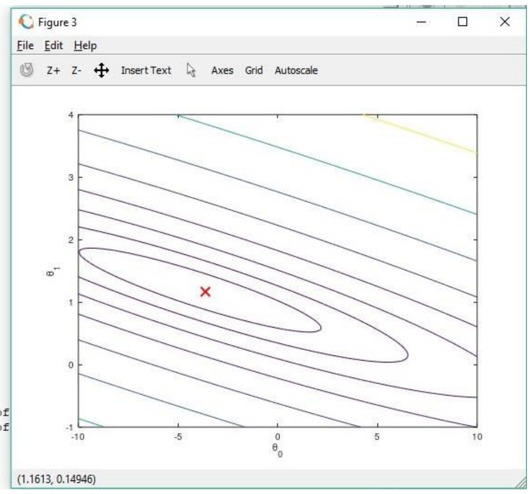
Setelah itu, langkah selanjutnya adalah membuat *surface* dan peta kontur. Hal ini dilakukan dengan code berikut ini.

warmUpExercise.m | ex1.m | plotData.m | computeCost.m | *gradientDescent.m

```
% initialize J_vals to a matrix of 0's
J_vals = zeros(length(theta0_vals), length(theta1_vals));

% Fill out J_vals
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


% Because of the way meshgrids work in the surf command, we need to
% transpose J_vals before calling surf, or else the axes will be flipped
J_vals = J_vals';
% Surface plot
figure;
surf(theta0_vals, theta1_vals, J_vals)
xlabel('\theta_0'); ylabel('\theta_1');

% Contour plot
figure;
% Plot J_vals as 15 contours spaced logarithmically between 0.01 and 100
contour(theta0_vals, theta1_vals, J_vals, logspace(-2, 3, 20))
xlabel('\theta_0'); ylabel('\theta_1');
hold on;
plot(theta(1), theta(2), 'rx', 'MarkerSize', 10, 'LineWidth', 2);
```

Ketika code dijalankan, hasil seperti di bawah ini pun akan ditampilkan.





BAB III

KESIMPULAN

Gradient descent bisa meminimalisasi hasil dari *cost function*. Hasil visualisasi menunjukkan bahwa perubahan pada gradien dengan *gradient descent* bisa memberikan nilai *global minimum*, yang merupakan hasil terkecil dari *cost function*.

Setelah mengerjakan tugas ini, kami semakin paham tentang implementasi *linear regression* dan *cost function*.

DAFTAR PUSTAKA

Suyanto. (November 2018). *Machine Learning: Tingkat Dasar dan Lanjut*. Bandung: Penerbit Informatika.