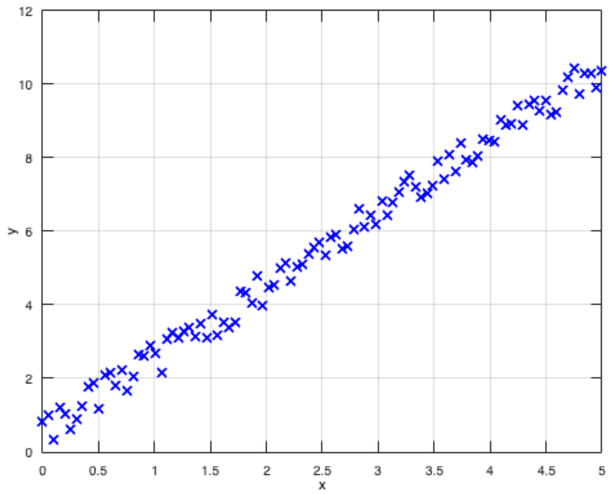
Multivariate Linear Regression

Ian Jasper A. Agulo, Ph.D.
Department of Physical Sciences
College of Science, University of the Philippines Baguio

Single Feature Linear Regression



When we started out this course, we used a single feature data set, such as the one shown. In this data set, the variable y is dependent only on the variable, x. Mathematically, we say that x is the independent variable and y is the dependent variable and it depends on the variable x. In machine learning, we call the variable x a feature of the variable y, because any changes to x will lead to a change in y.

I'm sure you can think of numerous examples of a single feature data set in physics. For example, the potential energy of an object is linearly dependent on its vertical height. The potential energy is proportionate to the height of your jump. Increase the mass of an object and its weight increases proportionately. For relatively low altitudes relative to sea level, the pressure decrease is also linearly proportional to the altitude.

There are, of course, variables, which sometimes depend on many other variables, we'll consider such an example next.

Multi-featured data set

	Acceleration	Cylinders	Displacement	Horsepower	MPG	Mfg	Model	Model_Year	Origin	Weight	cyl4
1	12	8	307	130	18	chevrolet	chevrolet chevelle malibu	70	USA	3504	Other
2	11.5000	8	350	165	15	buick	buick skylark 320	70	USA	3693	Other
3	11	8	318	150	18	plymouth	plymouth satellite	70	USA	3436	Other
4	12	8	304	150	16	amc	amc rebel sst	70	USA	3433	Other
5	10.5000	8	302	140	17	ford	ford torino	70	USA	3449	Other
6	10	8	429	198	15	ford	ford galaxie 500	70	USA	4341	Other
7	9	8	454	220	14	chevrolet	chevrolet impala	70	USA	4354	Other
8	8.5000	8	440	215	14	plymouth	plymouth fury iii	70	USA	4312	Other
9	10	8	455	225	14	pontiac	pontiac catalina	70	USA	4425	Other

	Acceleration	Cylinders	Displacement	Horsepower	MPG	Mfg	Model	Model_Year	Origin	Weight	cyl4
10	8.5000	8	390	190	15	amc	amc ambassador dpl	70	USA	3850	Other
11	17.5000	4	133	115	NaN	citroen	citroen ds-21 pallas	70	France	3090	Four
12	11.5000	8	350	165	NaN	chevrolet	chevrolet chevelle concours (...	70	USA	4142	Other
13	11	8	351	153	NaN	ford	ford torino (sw)	70	USA	4034	Other
14	10.5000	8	383	175	NaN	plymouth	plymouth satellite (sw)	70	USA	4166	Other
15	11	8	360	175	NaN	amc	amc rebel sst (sw)	70	USA	3850	Other
16	10	8	383	170	15	dodge	dodge challenger se	70	USA	3563	Other
17	8	8	340	160	14	plymouth	plymouth 'cuda 340	70	USA	3609	Other

This data set is a multi-featured data set. It describes the various features of various cars – its acceleration, the number of cylinders, the displacement of the cylinder, its horsepower, its weight, and its miles per gallon, among other features. To make it simple, we'll just select these columns with numerical values. Generally, we can use even the columns with text values by assigning a specific number to a specific value, but we won't do that for now. So, we'll only be considering the following variables – Acceleration, Cylinders, Displacement, Horsepower, Weight and MPG.

We will also take a look at how the MPG is affected by all the other variables. Thus, in this example, we'll consider MPG as the dependent variable and the remaining five variables as the independent variables or the features that the MPG is dependent on.

Before anything else, we need to clean the data. If we scroll down, we see that there are some cells with NaN or not a number values. So, we need to remove this. You'll see how this is done in the accompanying sample program. You'll also see that we removed the data set with texts.

Multi-featured data set

	Acceleration (x_1)	Cylinders (x_2)	Displacement (x_3)	Horsepower (x_4)	Weight (x_5)	MPG (y)
1	12	8	307	130	3504	18
2	11.5000	8	350	165	3693	15
3	11	8	318	150	3436	18
4	12	8	304	150	3433	16
5	10.5000	8	302	140	3449	17
6	10	8	429	198	4341	15
7	9	8	454	220	4354	14
8	8.5000	8	440	215	4312	14
9	10	8	455	225	4425	14

$$Single\ feature\colon\ h_\theta(x) = \theta_0 + \theta_1 x$$

$\sum_{i=0}^{n} \theta_i x_i$

$$Generalization\ of\ single\ feature\colon\ h_\theta(x) = \theta_0 x_0 + \theta_1 \underline{x_1}, x_0 = 1$$

Here's the clean and rearranged data set. The rows with element values of NaN have been removed. The first five columns are the features, while the last column is the dependent variable. We call the dependent variable as the hypothesis, the same way we called the y-axis in the single-featured data set the hypothesis. In the same way, we annotate our features as x_j, where j is the j^{th} feature and ranges from 1 to n, where n is the number of features. In other words, we call the Acceleration as x_1, the number of Cylinders as x_2, the Displacement as x_3, the Horsepower as x_4 and the Weight as x_5. There are $n = 5$ features.

Recall that for our single-featured data set, we have a linear equation relating the hypothesis with the feature. We can generalize this and rewrite the hypothesis as $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1$, where $x_0 = 1$. In machine learning, the features, x_j, are referred to as the input features, while the variable y is called the output feature.

Multi-featured data set

	x_1 Acceleration	x_2 Cylinders	x_3 Displacement	x_4 Horsepower	x_5 Weight	y MPG
1	12	8	307	130	3504	18
2	11.5000	8	350	165	3693	15
3	11	8	318	150	3436	18
4	$x_1^{(4)}$ $\boxed{12}$	8	304	150	3433	16
5	10.5000	8	302	140	3449	17
6	10	8	429	198	4341	15
7	9	8	454	220	4354	14
8	8.5000	8	440	215	4312	14
9	10	8	455	$x_4^{(9)}$ $\boxed{225}$	4425	14

$$Generalization\ of\ multi-feature:$$
$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5, \qquad x_0 = 1$$

Bias

For a multi-featured data set, we write $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5$, where $x_0 = 1$. Here, the Acceleration is labelled as x_1, the number of Cylinders as x_2, the Displacement as x_3, the Horsepower as x_4 and the Weight as x_5. Again, x_i are the input features. The MPG, denoted with y, is called the output feature.

To call on the row element, we use a superscript. In other words, the 4th element of the Acceleration is denoted as $x_1^{(4)}$. The 9th element of the Horsepower is denoted as $x_4^{(9)}$We enclose the row number with a parenthesis, so that it doesn't get confused with the power of a number.

We denote the row number with the variable, i, and the number of rows with the variable m, so that i$= 1\ to\ m$. This data set has 392 rows and 6 columns, so $m = 392$ and $n = 6$. Thus, we can represent this data set as a 392×6 matrix.

The idea here is to make a prediction of the best-fit line that minimizes the difference between the output feature, y, and the hypothesis, $h_\theta(x)$. Thus, the idea is to solve for the coefficients, θ_j, of the features, x_j. Recall that in the single-featured example, we solved for the coefficients θ_0 and θ_1. We called θ_0 the y-intercept, since, by definition, when all the coefficients of the features are zero, then $y = \theta_0$, considering that $x_0 = 1$. In the multi-featured case, θ_0, is still called the y-intercept, since the definition has not changed.

It's quite impossible to graph the multi-featured case. After all, we only live in a 3D world. How can we even begin to imagine a 6D plot? Nevertheless, we trust in the process. We use the same process in the single-featured data set for the multi-featured data set.

Gradient Descent for Multi-featured Data

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta\left(x^{(i)}\right) - y^{(i)}\right]^2$$

	X Data	Y Data
1	0	0.5590
2	0.0505	0.9551
3	0.1010	0.5499
4	0.1515	0.7491
5	0.2020	0.4583
6	0.2525	0.6822
7	0.3030	1.2689
8	0.3535	1.0379
9	0.4040	1.7066

X Data	Y Data	Error
x_1	y_1	e_1
x_2	y_2	e_2
x_3	y_3	e_3
$\vdots$	$\vdots$	$\vdots$
x_{n-2}	y_{n-2}	e_{n-2}
x_{n-1}	y_{n-1}	e_{n-1}
x_n	y_n	e_n

$$J(\theta_0, \theta_1) = \frac{1}{2n}\sum_{i=1}^{n}\left[h_\theta(x_i) - y_i\right]^2$$

We begin with the hypothesis. This is a generalized version. The subscript is denoted by the variable $j = 0\ to\ n$ and n is the number of input features. We write the cost function as

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta\left(x^{(i)}\right) - y^{(i)}\right]^2$$

When you review your notes for the single-featured cost function, you'll probably see this equation

$$J(\theta_0, \theta_1) = \frac{1}{2n}\sum_{i=1}^{n}\left[h_\theta(x_i) - y_i\right]^2$$

They don't look alike, but they represent exactly the same thing. Recall that in the single-featured data set, you take the difference of the hypothesis, $h_\theta(x)$, and the output feature, y, per row, i, square each of them, and then add them all up before dividing everything by $2m$.

Gradient Descent for Multi-featured Data

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta\left(x^{(i)}\right) - y^{(i)}\right]^2$$

	x_1	x_2	x_3	x_4	x_5	y
	Acceleration	Cylinders	Displacement	Horsepower	Weight	MPG
1	12	8	307	130	3504	18
2	11.5000	8	350	165	3693	15
3	11	8	318	150	3436	18
4	12	8	304	150	3433	16
5	10.5000	8	302	140	3449	17

Here, we are doing exactly the same thing. You take the difference of the hypothesis, $h_\theta(x)$, and the output feature, y, per row, i, square each of them, and then add them all up before dividing everything by $2m$, not $2n$, because we recently denoted the number of rows as m. We also made a changed in the notation of the row number from a subscript to a superscript enclosed with a parenthesis. Nevertheless, they both represent the same physical quantity, but this one is more generalized to n number of features.

Gradient Descent for Multi-featured Data

Derivatives of the Cost Function:

$$\frac{\partial J(\theta_0,\theta_1)}{\partial\theta_0} = \frac{1}{n}\sum_{i=1}^{n}[(\theta_0+\theta_1 x_i)-y_i]$$

$$\frac{\partial J(\theta_0,\theta_1)}{\partial\theta_1} = \frac{1}{n}\sum_{i=1}^{n}[(\theta_0+\theta_1 x_i)-y_i]x_i$$

Derivatives of the Cost Function:

$$\frac{\partial J(\theta_0,\theta_1)}{\partial\theta_0} = \frac{1}{m}\sum_{i=1}^{m}\left[\left(\theta_0+\theta_1 x^{(i)}\right)-y^{(i)}\right]$$

$$\frac{\partial J(\theta_0,\theta_1)}{\partial\theta_1} = \frac{1}{m}\sum_{i=1}^{m}\left[\left(\theta_0+\theta_1 x^{(i)}\right)-y^{(i)}\right]x^{(i)}$$

Recall that for the single-featured data set, we derived the derivatives of the cost function as these two expressions.

$$\frac{\partial J(\theta_0,\theta_1)}{\partial\theta_0} = \frac{1}{m}\sum_{j=1}^{m}[(\theta_0+\theta_1 x_i)-y_i] \quad \text{and} \quad \frac{\partial J(\theta_0,\theta_1)}{\partial\theta_1} = \frac{1}{n}\sum_{i=1}^{n}[(\theta_0+\theta_1 x_i)-y_i]x_i$$

In our new notation, this is rewritten as

$$\frac{\partial J(\theta_0,\theta_1)}{\partial\theta_0} = \frac{1}{m}\sum_{i=1}^{m}\left[\left(\theta_0+\theta_1 x^{(i)}\right)-y^{(i)}\right] \quad and \quad \frac{\partial J(\theta_0,\theta_1)}{\partial\theta_1} = \frac{1}{m}\sum_{i=1}^{m}\left[\left(\theta_0+\theta_1 x^{(i)}\right)-y^{(i)}\right]x^{(i)}$$

Gradient Descent for Multi-featured Data

Derivatives of the Cost Function:

$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0} = \frac{1}{m}\sum_{j=1}^{m}\left[\left(\theta_0 + \theta_1 x^{(i)}\right) - y^{(i)}\right]$$

$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1} = \frac{1}{m}\sum_{j=1}^{m}\left[\left(\theta_0 + \theta_1 x^{(i)}\right) - y^{(i)}\right]x^{(i)}$$

Generalization:

$$\frac{\partial J(\theta_0, \theta_1, \ldots, \theta_n)}{\partial \theta_i} = \frac{1}{m}\sum_{i=1}^{m}\left[\sum_{j=1}^{n}\theta_j x_j^{(i)} - y^{(i)}\right]x_j^{(i)}, x_{j=0}^{(i)} = 1$$

We can generalize this, in fact, into one single equation, given by

$$\frac{\partial J(\theta_0, \theta_1, \ldots, \theta_n)}{\partial \theta_i} = \frac{1}{m}\sum_{i=1}^{m}\left[\sum_{j=1}^{n}\theta_j x_j^{(i)} - y^{(i)}\right]x_j^{(i)}$$

When $j = 0, x_j^{(i)} = 1$.

Gradient Descent for Multi-featured Data

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta\left(x^{(i)}\right) - y^{(i)}\right]^2$$

$$\frac{\partial J(\theta_0, \theta_1, \ldots, \theta_n)}{\partial \theta_i} = \frac{1}{m}\sum_{i=1}^{m}\left[\sum_{j=1}^{n}\theta_j x_j^{(i)} - y^{(i)}\right] x_j^{(i)}, x_{j=0}^{(i)} = 1$$

$$iterate\ \{\theta_j = \theta_j - \alpha\frac{\partial J(\theta_0, \theta_1, \ldots, \theta_n)}{\partial \theta_j}, j = 0,1,2, \ldots, n\}$$

We need only one more step to implement gradient descent and that is to iteratively change the feature values by subtracting the computed gradient descent during each iteration.

In this slide, you have everything you need to implement machine learning.

Implementation of Gradient Descent

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

	x_0	x_1	x_2	x_3	x_4	x_5	y
	Bias	Acceleration	Cylinders	Displacement	Horsepower	Weight	MPG
1	1	12	8	307	130	3504	18
2	1	11.5000	8	350	165	3693	15
3	1	11	8	318	150	3436	18
4	1	12	8	304	150	3433	16
5	1	10.5000	8	302	140	3449	17
6	1	10	8	429	198	4341	15
7	1	9	8	454	220	4354	14
8	1	8.5000	8	440	215	4312	14
9	1	10	8	455	225	4425	14

Now, it's time to implement this in code.

Let's begin with the hypothesis. First, let's add a column to include x_0. The column matrix all have values of 1, as shown. We now have six input features, including the bias, and one output feature.

Implementation of Gradient Descent

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad h_\theta(x) = \theta^T x = [\theta_0 \quad \theta_1 \quad \cdots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

We're going to rewrite the hypothesis, $h_\theta(x)$, in matrix notation. We write both the coefficients and features as column matrices, θ and x, respectively.

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \; and \;\; x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Then, we take the transpose of matrix, θ, and multiply it by the matrix, x. So, that our new hypothesis is in the form of a matrix, given by

$$h_\theta(x) = \theta^T x = [\theta_0 \quad \theta_1 \quad \cdots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

This is the mathematical operation for each row.

$$h_\theta(x) = \theta^T x = [\theta_0 \quad \theta_1 \quad \cdots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$[\theta_0 \quad \theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad]$

ans = 7×392 table

		cars_temp1	cars_temp2	cars_temp3	cars_temp4
1	**Bias**	1	1	1	1
2	**Acceleration**	-1.2836	-1.4649	-1.6461	-1.2836
3	**Cylinders**	1.4821	1.4821	1.4821	1.4821
4	**Displacement**	1.0759	1.4868	1.1810	1.0472
5	**Horsepower**	0.6633	1.5726	1.1829	1.1829
6	**Weight**	0.6197	0.8423	0.5397	0.5362
7	**MPG**	-0.6977	-1.0821	-0.6977	-0.9540

This is the mathematical operation for each row. We note that data set has been transposed. The output of multiplying the row matrix θ with $1{\times}6$ dimensions with the whole data set for x with $6{\times}392$ dimension is a hypothesis that is a row matrix with $1{\times}392$ dimension.

The matrix y, which is the 7th row in the transposed data set, is also a

$$h_\theta(x) = \theta^T x = [\theta_0 \quad \theta_1 \quad \cdots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$h_\theta(x) - y = \theta^T x - y = [\theta_0 \quad \theta_1 \quad \cdots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} y_0 \\ y_0 \\ \vdots \\ y_n \end{bmatrix}$$

To get the cost-function, we first subtract the output feature, y, from the hypothesis, where the column, y, is a column matrix, given by

$$y = \begin{bmatrix} y_0 \\ y_0 \\ \vdots \\ y_n \end{bmatrix}$$

Thus, the difference is now written as

$$h_\theta(x) - y = \theta^T x - y = [\theta_0 \quad \theta_1 \quad \cdots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} y_0 \\ y_0 \\ \vdots \\ y_n \end{bmatrix}$$

Then, we square this difference and sum the squares before dividing this by twice the number of rows to get the cost function.

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}\left[\boxed{h_\theta\left(x^{(i)}\right)} - \boxed{y^{(i)}}\right]^2$$

$[\theta_0 \quad \theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5 \quad \theta_6]$

ans = 7×392 table

		cars_temp1	cars_temp2	cars_temp3	cars_temp4
1	Bias	1	1	1	1
2	Acceleration	-1.2836	-1.4649	-1.6461	-1.2836
3	Cylinders	1.4821	1.4821	1.4821	1.4821
4	Displacement	1.0759	1.4868	1.1810	1.0472
5	Horsepower	0.6633	1.5726	1.1829	1.1829
6	Weight	0.6197	0.8423	0.5397	0.5362
7	MPG	-0.6977	-1.0821	-0.6977	-0.9540

Thus, when solving for the cost function, the hypothesis, $h_\theta\left(x^{(j)}\right)$, is a row matrix with $1\times m$ dimension, where $m = 392$, the number of rows of the data set.

The matrix y, which is the 7$^{\text{th}}$ row in the transposed data set, is also a row matrix with $1\times m$ dimension.

Thus, the difference $h_\theta\left(x^{(i)}\right) - y^{(i)}$ is also a row matrix with $1\times m$ dimension. When we sum over all rows and divide by $2m$, we get a single value for the cost function.

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta\left(x^{(i)}\right) - y^{(i)}\right]^2 \qquad h_\theta(x) = \theta^T x$$

```
theta'*x'-y'
(x*theta)'-y'
(x*theta-y)'
```

```
sum((x*theta-y)'.^2)
sum((x*theta-y).^2)
```

```
function [Jcost] = costfunction(x,y,theta)
    m = length(y);
    Jcost = (1/(2*m))*sum((x*theta-y).^2);
end
```

```
def costfunction(x,y,theta):
  m = len(y)
  Jcost = (1/(2*m))*np.sum(np.power(
        (np.dot(x,theta)-y.reshape(m,1)),2))
  return Jcost
```

Here's how we implement this in code.

First, let's take a look at the hypothesis. Recall that the hypothesis is written as

$$h_\theta(x) = \theta^T x$$

Recall as well that the input feature variable x is a 392×6 matrix and needs to be transposed so that it is correctly multiplied with θ^T which is a 1×6 matrix. The product gives us a 1×392 matrix. Since the output feature variable y is a 392×1 matrix, it needs to be transposed as well to be able to perform proper matrix subtraction. The difference still gives us a 1×392 matrix. Rewriting $theta' * x'$ as $(x * theta)'$ will make it easier for us to transpose the whole expression. This becomes then a 392×1 matrix.

Then, we take the square of the transpose of the difference between $x * theta$ and y and the sum all terms. This is the same as with not doing the transpose of the difference, since we're just taking the sum of all terms.

Thus, we have the final expression in code for the cost function written as a MATLAB function $costfunction$ with input arguments x, y, and $theta$. For Python, it's practically the same. The only difference is that the output feature variable y is reshaped from $(m,)$ to $(m,1)$, so that it becomes compatible with the shape of $x * theta$.

Implementation of Gradient Descent

$$\frac{\partial J(\theta_0, \theta_1, \dots, \theta_n)}{\partial \theta_i} = \frac{1}{m} \sum_{i=1}^{m} \left[\sum_{j=1}^{n} \theta_j x_j^{(i)} - y^{(i)} \right] x_j^{(i)}, x_{j=0}^{(i)} = 1$$

$$iterate \left\{ \theta_j = \theta_j - \alpha \frac{\partial J(\theta_0, \theta_1, \dots, \theta_n)}{\partial \theta_j}, j = 0,1,2, \dots, n \right\}$$

```
for k = 1:epoch
    grad_theta = (1/m)*x'*(x*theta-y);
    theta = theta - a*grad_theta;
    J(k) = costfunction(x,y,theta);
end
```

```
for k in range(epoch):
    grad_theta = (1/m)*np.dot(
        np.transpose(x),((np.dot(x,theta)-y.reshape(m,1))))
    theta = theta - a*grad_theta
    J[k] = costfunction(x,y,theta)
```

Next, we rewrite the gradient in code. We know that the term in the brackets 392×1 matrix. Each of these terms is multiplied by the term $x_j^{(i)}$, which represents the j^{th} row of the i^{th} column of the input feature x. Thus, we are performing a matrix multiplication of a 392×1 matrix by a 392×1 matrix. The easiest is to transpose $x^{(i)}$and multiply this by the term in brackets.

Thus, this is a dot product of the transpose of $x^{(i)}$ and the term in brackets. The product gives us a single term, which is the gradient of $J(\theta_0, \theta_1, \dots, \theta_n)$ with respect to θ_i. We can conveniently perform a dot product of the transpose of x, which is a 6×392, and the term in brackets, which gives us all the elements of the gradient for all θ_i. The output is a 6×1 matrix corresponding the gradient of J with respect to all θ.

This is then subtracted from the previous value of θ to get new values of θ. The process is repeated for the given number of epochs.

The cost function is also computed in each iteration and plotted to get a sense of whether the gradient descent method performed well.

Let’s take a look at how this is implemented in MATLAB and Python.

Feature Scaling

By rescaling the input features to an approximate range (or in the order) of $-1 \leq x \leq 1$, then cost function converges faster towards the optimum value.

$$X = \frac{X - \bar{X}}{\Delta X}$$

```
function [Xfeat] = FeatureScale(X)
    Xmean = mean(X);
    Xstd = std(X);
    Xfeat = (X - Xmean)./(Xstd);
end
```

```
def FeatureScale(X):
  Xmean = np.mean(X)
  Xstd = np.std(X)
  Xfeat = (X - Xmean)/Xstd
  return Xfeat
```

Feature scaling is a way to improve the efficiency of gradient descent and allowing it to converge faster towards the minimum value. This is a quite simple method. The scaled values are obtained by subtracting the mean value of the input features to the input features and then dividing the result by the standard deviation of the input features.

Summary

- Supervised vs unsupervised machine learning
- Univariate and Multivariate Linear Regression
- Gradient Descent
- Machine Learning Features, Theta
- Machine Learning Parameters
- Machine Learning Terms
  - Epoch
- Hyperparameters
  - Learning Rate
- Techniques to Improve Accuracy
  - Feature Scaling