# Univariate Linear Regression

Regression-Based Supervised Learning

*Using the Example of a House Price Predictor*

## Terms Used:

### Training Set:

This is the data which is used to teach the computer what type of outputs we are looking for.

In this case, the training set we use is the size of the house in sq. ft. with its corresponding cost.

|  |  |
| --- | --- |
| Size (feet2) | Cost ($) |
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| … | … |

The number of training examples in the training set is denoted by ‘m’. **m = 1460**

### Hypothesis Function:

This is the prediction which we make that maps values of x to values of y. In this case, it maps from the house area to its cost. The better the hypothesis function, the closer the predicted value is to the actual value.

In this case, the better the hypothesis function, the closer the value of the predicted price to the actual price of the house.

### Cost Function:

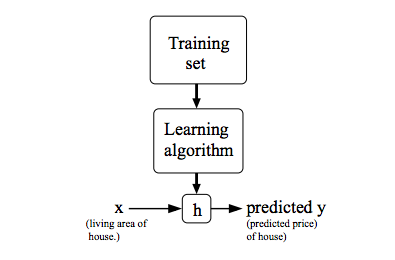
This function checks how error-prone a certain hypothesis function is. The lower the value of the cost function, the closer the predicted value is to the actual value.

In this case, the lower the value of the cost function, the closer the value of the predicted price to the actual price of the house.

## Notation:

|  |  |
| --- | --- |
| Notation used | Meaning |
| m | Number of training examples |
| x | Input variables (features) (Lot Area in this case) |
| y | Output values (target) (Selling price of house in this case) |
| (x, y) | A random training example |
| (x(i), y(i)) | The ith training example (e.g. (x (2), y (2)) = (1543, 315)) |
| *h* | Hypothesis function (prediction function, which maps from x to y) |
| *J* | Cost function |

## Flow of Control:



## Representation of *h:*

Normal Representation:

Shorthand Representation:

()

## Process:

Find a hypothesis function such that the value of:

is minimized. (i.e. find the values of θ0 and θ1 such that the value of is minimized. This is mathematically denoted as:

This function, , is called the cost function. (It is also called the squared error function or the squared error cost function.)

If any one of the parameters, , are fixed, then the problem becomes a minimization with respect to only one of the variables, as the other is fixed.

### How it works (Intuition of the Cost Function):

The Cost function reaches its minima (least value) for a certain value of . This value of results in the line that passes through nearly the middle of the scattered data.

This line that we get by minimizing the Cost Function is the best possible hypothesis that we can use to predict future results.

### Minimizing the Cost Function (J) using Gradient Descent:

Gradient Descent is a method in which we move down from one point to another point in the direction of maximum gradient from any point to a local or global minimum.

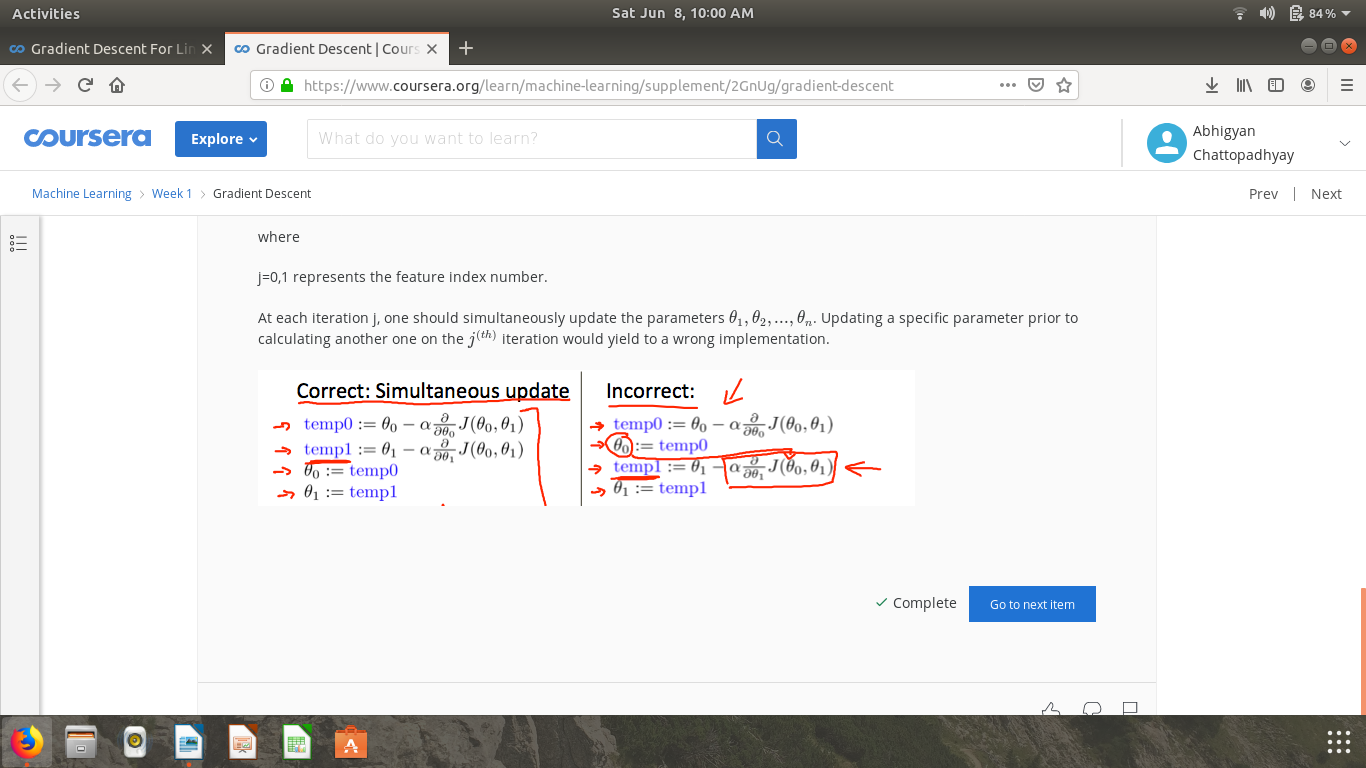
Aside: The gradient descent can be applied to a large variety of functions, having infinitely many inputs like (), not necessarily just 2 like we have here ().

This is achieved by using the equation:

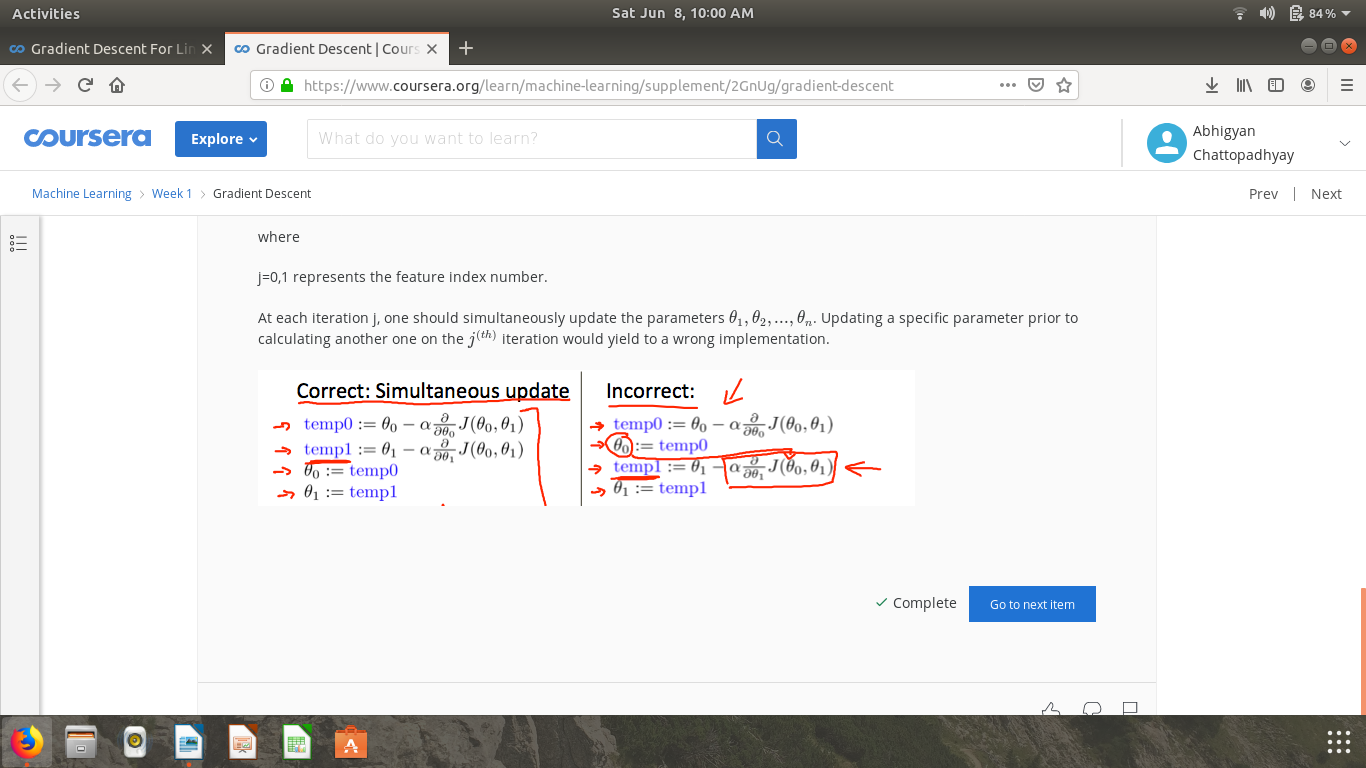
, where i can be 0 or 1.

Here, the value is called the “Learning Rate”.

Now, the correct way to update the value of is called “Simultaneous Update”, which works as follows:



It should not be used in any other way, as any other way is not considered to be a simultaneous update, for example:



The Learning rate has some special properties:

1. If is too large, then the gradient descent algorithm may not converge, or even diverge.
2. If is too small, then the gradient descent is too slow, as a result of which the algorithm runs too slowly.

Also, as the value of or reaches a local minimum, the value of automatically takes on smaller values, and its sign (positive or negative) changes to make the value of or increase or decrease as required, and eventually stop changing, when it becomes zero. This happens because the value of becomes zero when the slope of the cost function becomes zero, that is, when the cost function reaches a local minima.

When this is applied to a linear regression, whose is given by , we can evaluate the value of and thus find in each case (i=0 and 1).

It turns out that

and

(Proof in appendix)

### Issues with Gradient Descent:

Gradient descent is susceptible to giving local minima instead of global minima. This is not a problem in case of convex functions due to them having only one local minima, such as the cost function of a linear regression.

However, when we deal with non-linear regressions, we may come across cases where the gradient descent algorithm results in local minima instead of the global minima.

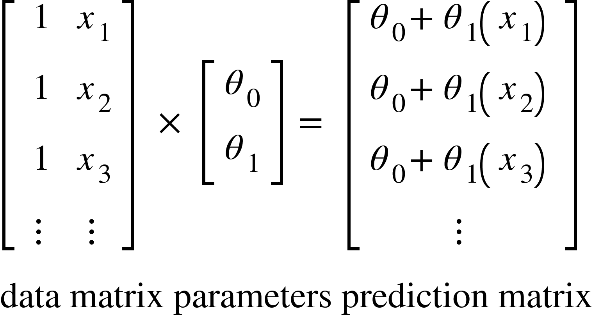
# Linear Algebra tricks for simplifying Machine Learning (Vectorization)

*Using the Example of a House Price Predictor*

Assuming that Matrix Multiplication is already available in our libraries/toolkit, we can easily use it for various purposes:

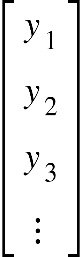
## 1. To check one hypothesis for different training examples:

It is done as follows:



The prediction matrix is a n×1 matrix which contains the values of the hypothesis for each value of input present in the second column of the data matrix.

This can be easily compared with another matrix which contains the values of y, i.e.



## 2. To check which hypothesis is better:

If the parameters of each hypothesis is represented as , then the following matrix product gives a matrix whose ith column gives the value of the ith hypothesis evaluated for the data set in its respective row, i.e.:

<math xmlns="http://www.w3.org/1998/Math/MathML"><mfenced open="[" close="]"><mtable><mtr><mtd><mn>1</mn></mtd><mtd><msub><mi>x</mi><mn>1</mn></msub></mtd></mtr><mtr><mtd><mn>1</mn></mtd><mtd><msub><mi>x</mi><mn>2</mn></msub></mtd></mtr><mtr><mtd><mn>1</mn></mtd><mtd><msub><mi>x</mi><mn>3</mn></msub></mtd></mtr><mtr><mtd><mo>&#x22EE;</mo></mtd><mtd><mo>&#x22EE;</mo></mtd></mtr></mtable></mfenced><mo>&#xD7;</mo><mfenced open="[" close="]"><mtable><mtr><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>1</mn></msubsup></mtd><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>2</mn></msubsup></mtd><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>3</mn></msubsup></mtd><mtd><mo>&#x22EF;</mo></mtd></mtr><mtr><mtd><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>1</mn></msubsup></mtd><mtd><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>2</mn></msubsup></mtd><mtd><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>3</mn></msubsup></mtd><mtd><mo>&#x22EF;</mo></mtd></mtr></mtable></mfenced><mo>=</mo><mfenced open="[" close="]"><mtable><mtr><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>1</mn></msubsup><mo>+</mo><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>1</mn></msubsup><msub><mi>x</mi><mn>1</mn></msub></mtd><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>2</mn></msubsup><mo>+</mo><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>2</mn></msubsup><msub><mi>x</mi><mn>1</mn></msub></mtd><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>3</mn></msubsup><mo>+</mo><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>3</mn></msubsup><msub><mi>x</mi><mn>1</mn></msub></mtd><mtd><mo>&#x22EF;</mo></mtd></mtr><mtr><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>1</mn></msubsup><mo>+</mo><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>1</mn></msubsup><msub><mi>x</mi><mn>2</mn></msub></mtd><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>2</mn></msubsup><mo>+</mo><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>2</mn></msubsup><msub><mi>x</mi><mn>2</mn></msub></mtd><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>3</mn></msubsup><mo>+</mo><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>3</mn></msubsup><msub><mi>x</mi><mn>2</mn></msub></mtd><mtd><mo>&#x22EF;</mo></mtd></mtr><mtr><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>1</mn></msubsup><mo>+</mo><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>1</mn></msubsup><msub><mi>x</mi><mn>3</mn></msub></mtd><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>3</mn></msubsup><mo>+</mo><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>3</mn></msubsup><msub><mi>x</mi><mn>3</mn></msub></mtd><mtd><msubsup><mi>&#x3B8;</mi><mn>0</mn><mn>3</mn></msubsup><mo>+</mo><msubsup><mi>&#x3B8;</mi><mn>1</mn><mn>3</mn></msubsup><msub><mi>x</mi><mn>3</mn></msub></mtd><mtd><mo>&#x22EF;</mo></mtd></mtr><mtr><mtd><mo>&#x22EE;</mo></mtd><mtd><mo>&#x22EE;</mo></mtd><mtd><mo>&#x22EE;</mo></mtd><mtd><mo>&#x22F1;</mo></mtd></mtr></mtable></mfenced></math>

Thus, each entry of the resultant matrix is the value of the respective hypothesis evaluated for the respective data entry.

On comparing this resultant matrix with the values of y, we can get an idea of which hypothesis is better for the given data set.

# Multivariate Linear Regression

Regression-Based Supervised Learning

Here, we deal with more than one feature to make a prediction.

For example, if we used the size of the house in sq. ft., number of floors, number of bedrooms and the age of the house, all of these as features to determine the price of the house instead of just its size, then the problem is considered as a Multivariate Linear Regression.

Here is some sample data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Size (feet2) | Number of Bedrooms | Number of floors | Age of Home (years) | Cost ($1000) |
| 2104 | 5 | 1 | 45 | 460 |
| 1416 | 3 | 2 | 40 | 232 |
| 1534 | 3 | 2 | 30 | 315 |
| 852 | 2 | 1 | 36 | 178 |
| … | … | … | … | … |

## New Notation:

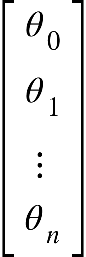
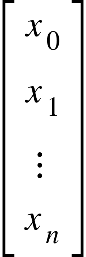
This uses some different notation from the univariate linear regression, i.e.:

|  |  |
| --- | --- |
| Notation used | Meaning |
| m | Number of training examples |
| xj | The jth feature (j<=n) |
| y | The output value (target) |
| (x1(i), x2(i), x3(i), x4(i), …, y(i)) | The ith training example |
| *h* | Hypothesis function (prediction function, which maps from x to y) |
| *J* | Cost Function |
| n | Number of features (here, it is 4) |

As a result, our hypothesis function has a new form:

Now, if we assume some x0 which is always equal to 1, then we may also write the above equation as:

This can again be simplified using some Linear Algebra:

If we define a some vectors  and X=, then our work is simplified tremendously.

Now, we can express , thus simplifying our notation and our working. (Here, represents the transpose of matrix )

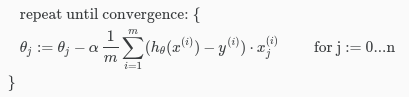
## New Cost Function and partial derivatives:

The new cost function is given by:

Now, the new algorithm of gradient descent contains many partial derivative terms, one for each . However, on simplifying the partial derivative for any term θj, the value of .

Aside: This is similar to how , where = 1 and . Thus, the complete formula actually remains the same as it was before.

Thus, the final gradient descent algorithm for any is given by:



# Practical Improvements for Gradient Descent:

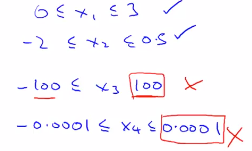
## Feature Scaling:

If the features we measure are of very different ranges, then our contours become extremely skewed (read skinny and long ovals), which makes gradient descent quite lengthy, due to them taking on long and meandering paths (i.e. zig-zag paths). Overall, this leads to slow-paced algorithms which often don’t work well.

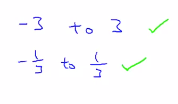
When we scale features, our contours work much better due to circular contours which are easier for the gradient descent to descend.

In order to scale features properly, we should try to divide or multiply our features such that their new range is approximately .

### Examples:



### Good practices:

According to the course instructor, Andrew Ng, these are some ranges beyond which feature scaling is required.

Anything between is fine.

Anything beyond either of these should be scaled down in the first case and scaled up in the second. However, this is not an absolute limit, and even ranges beyond these can be used provided that the algorithm is working smoothly and quickly enough.

## Mean Normalisation:

Disclaimer: This is not to be applied to the

This is the process of replacing the features with in order to make the features’ mean approximately zero.

Here, represents the mean of each feature.

## Feature Scaling + Mean Normalization:

This is the process of replacing the features with in order to make the features’ mean approximately zero and also bring the values between the range of -0.5 to +0.5.

Here, represents the mean of each feature, and represents the range of values taken on by (i.e. max value - min value).

## Debugging:

### Check if is decreasing with the number of iterations

The value of should either decrease as the number of iterations increases, or remain the same in case we have initially started at the minima itself.

It is best to plot vs number of iterations before we make any assumptions.

Ifincreases with the number of iterations, then the gradient descent is not working correctly. In this case, we need to use a smaller learning rate, untilstarts decreasing irrespective of the initial value.

However, ifremains constant with number of iterations, then it may be possible that we have chosen the minima ofas our initial value, or that the gradient descent is not working. In this case, we need to change the initial value so that we can check whether this occurs only at one initial value or at all values.

In case this occurs for all initial values, then the gradient descent is not working properly, and the learning rate may be zero. Hence, we now need to use some positive value of learning rate.

If goes down and then goes up, even then we need to use a smaller value of learning rate to correct our algorithm.

### Deciding thathas converged

We can only say that has converged when its value doesn’t change very significantly with time. However, the definition of ‘significantly’ is variable for different applications. Hence, we need to take a reasonably small value depending on the scale we are using, and then declare convergence in case changes by a value less than .

It is better to choose after looking at a plot of vs number of iterations, rather than randomly choosing a threshold value.