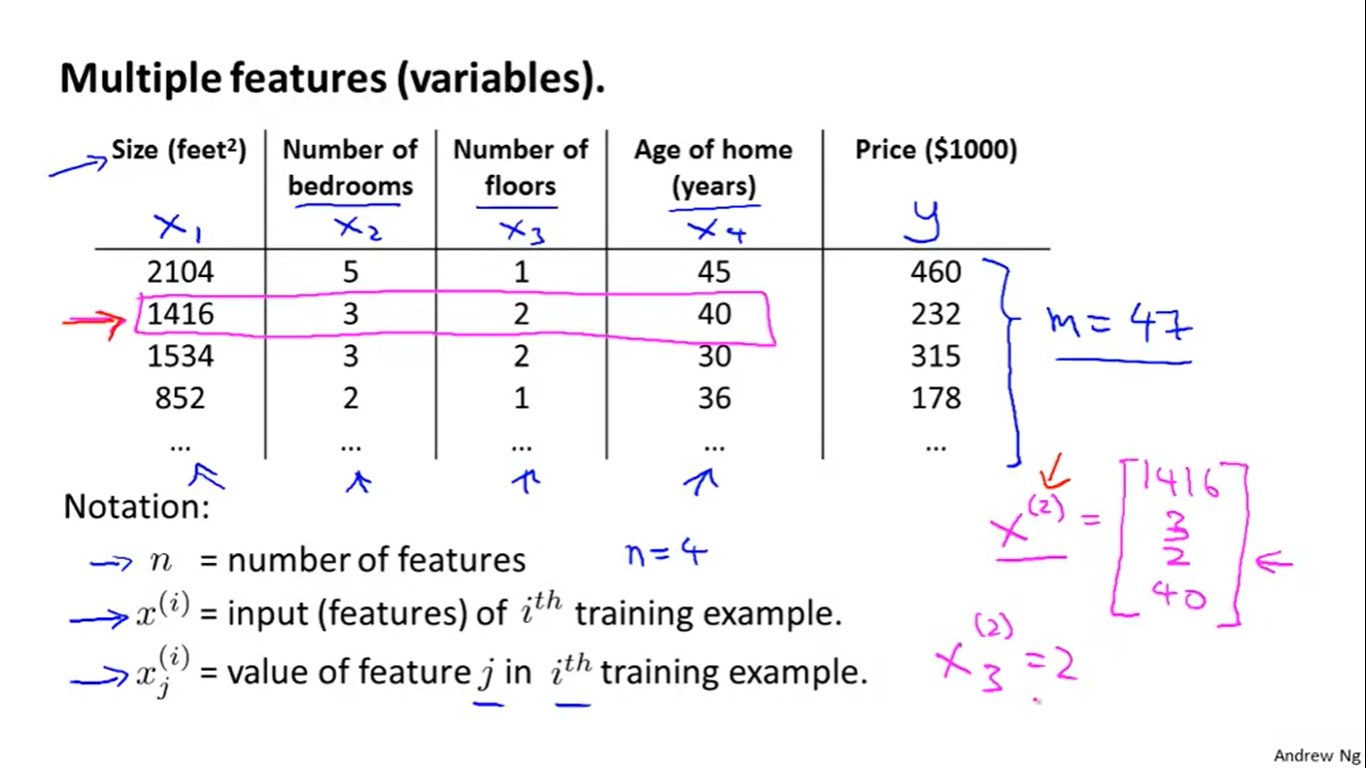
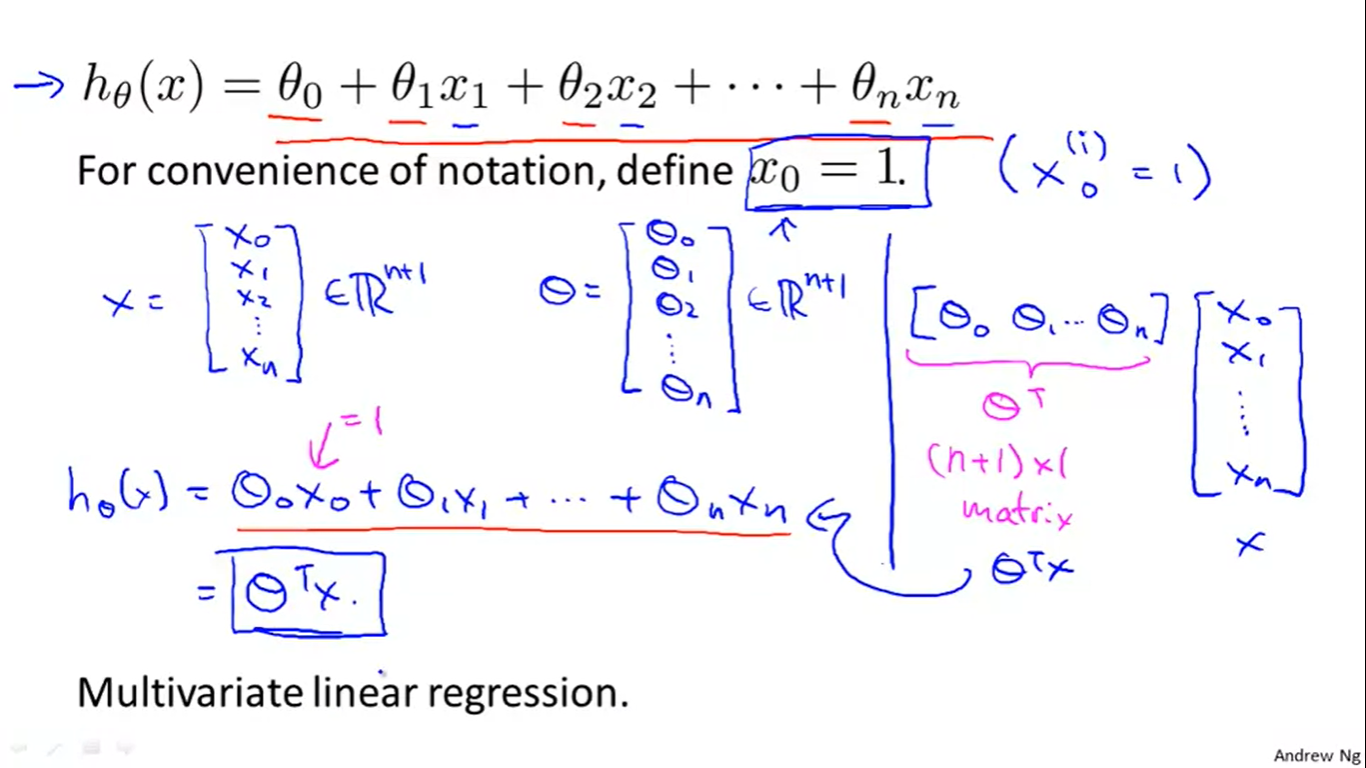
Multiple features :

Linear regression with multiple variables is also known as "multivariate linear regression".



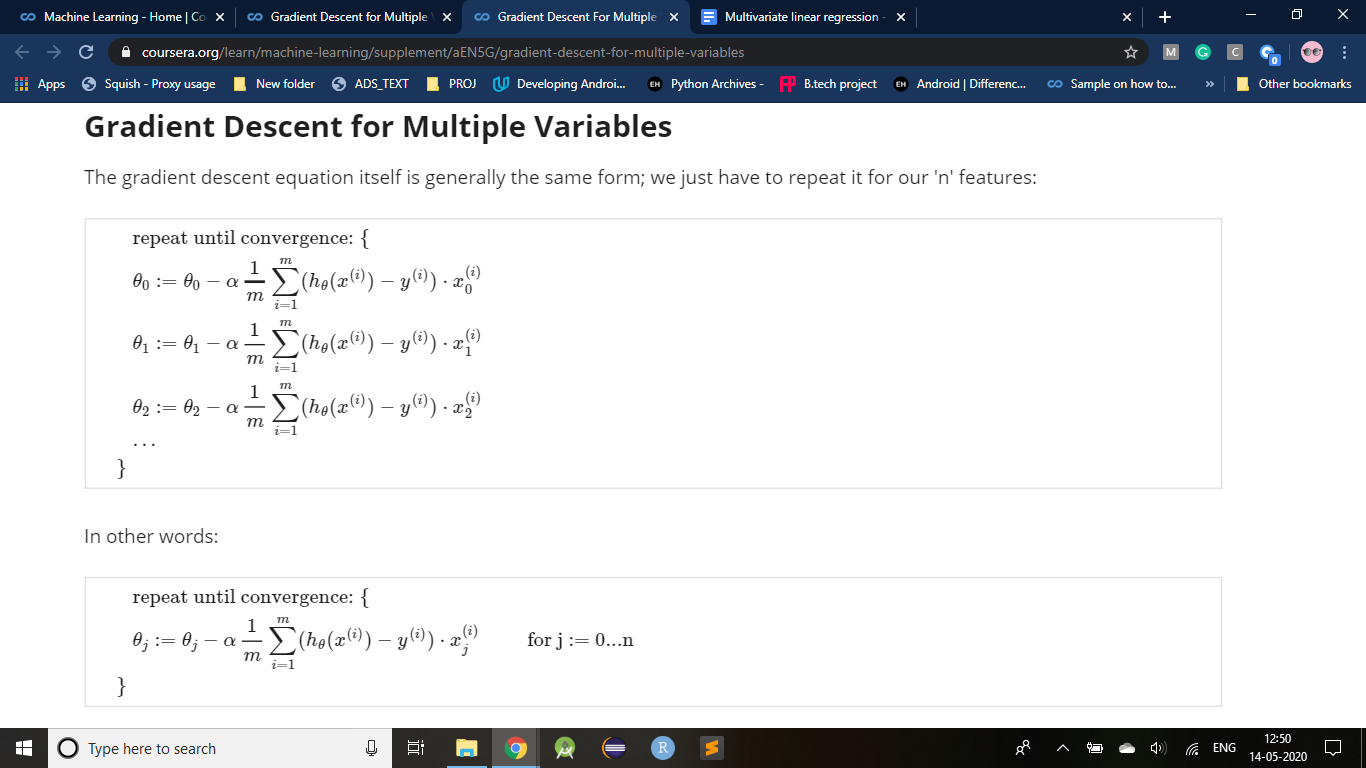
In order to develop intuition about this function, we can think about *θ*0as the basic price of a house, *θ*1as the price per square meter, *θ*2 as the price per floor, etc. x1 will be the number of square meters in the house, x2 the number of floors, etc.

Using the definition of matrix multiplication, our multivariable hypothesis function can be concisely represented as:



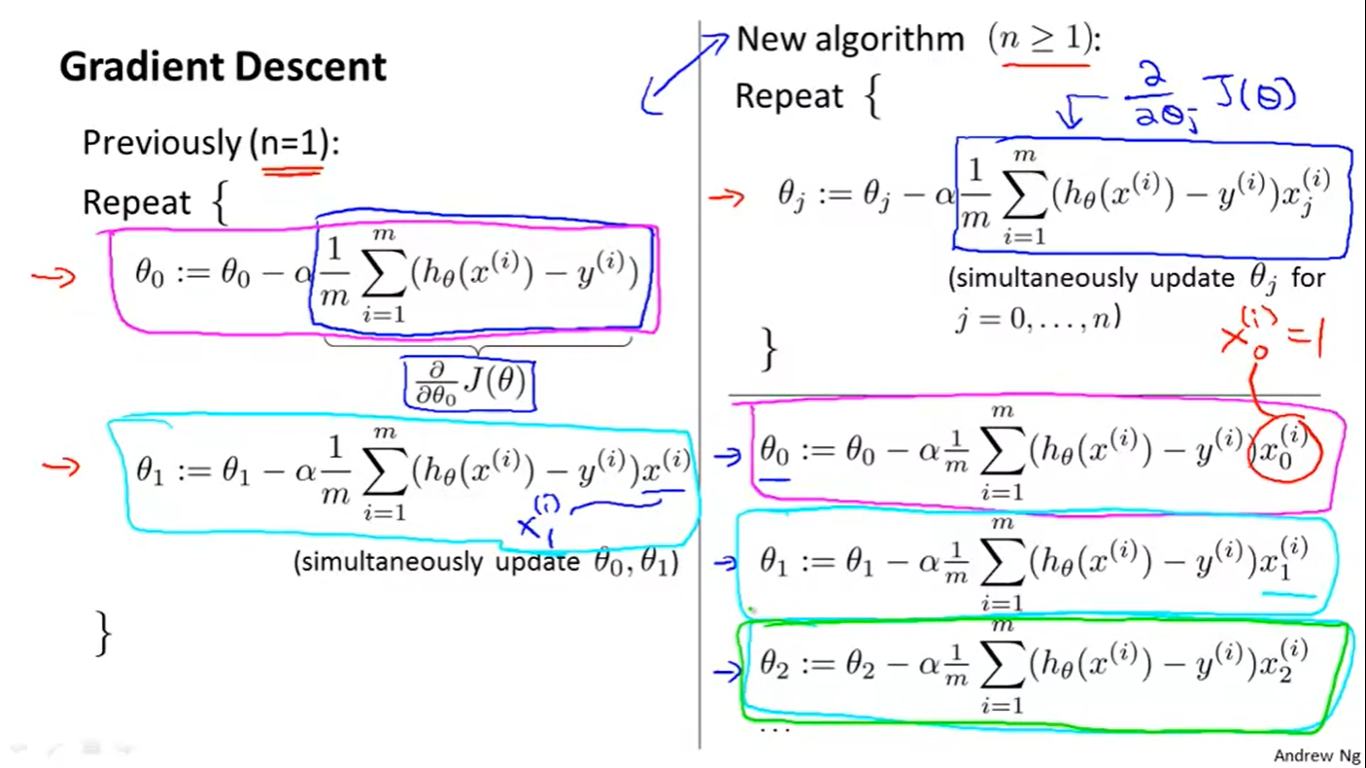
For matrix operations(and convenience) x0 = 1.

#### Gradient Descent for Multiple Variables :



The following image compares gradient descent with one variable to gradient descent with

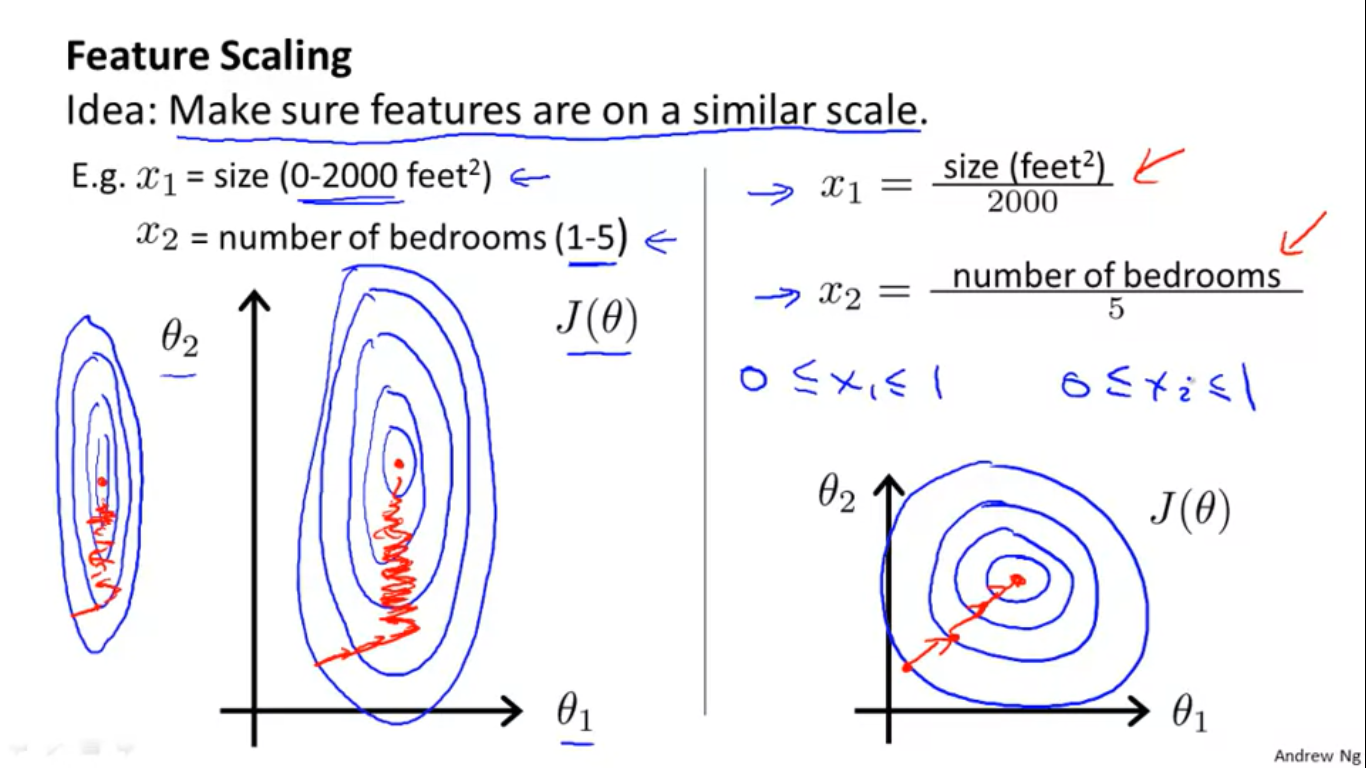
multiple variables:



#### Gradient Descent in Practice I - Feature Scaling :

We can speed up gradient descent by having each of our input values in roughly the same range. This is because θ will descend quickly on small ranges and slowly on large ranges, and so will oscillate inefficiently down to the optimum when the variables are very uneven.

Idea : make sure features are on similar scale so that gradient descent can converge(to global optimum) more quickly.



The elliptical contours may take really long to find the global optimum(crossing the whole path) as they are really skewed and the 2000:5 would look even more skewed.Gradient descent may have to go back and forth to find global optimum.

Whereas,

In case 2 when the features are brought to the same scale (0 to 1) they may lie in approximately similar ranges so gradient descent would take less time and the contour would look something of that kind in fig 2.

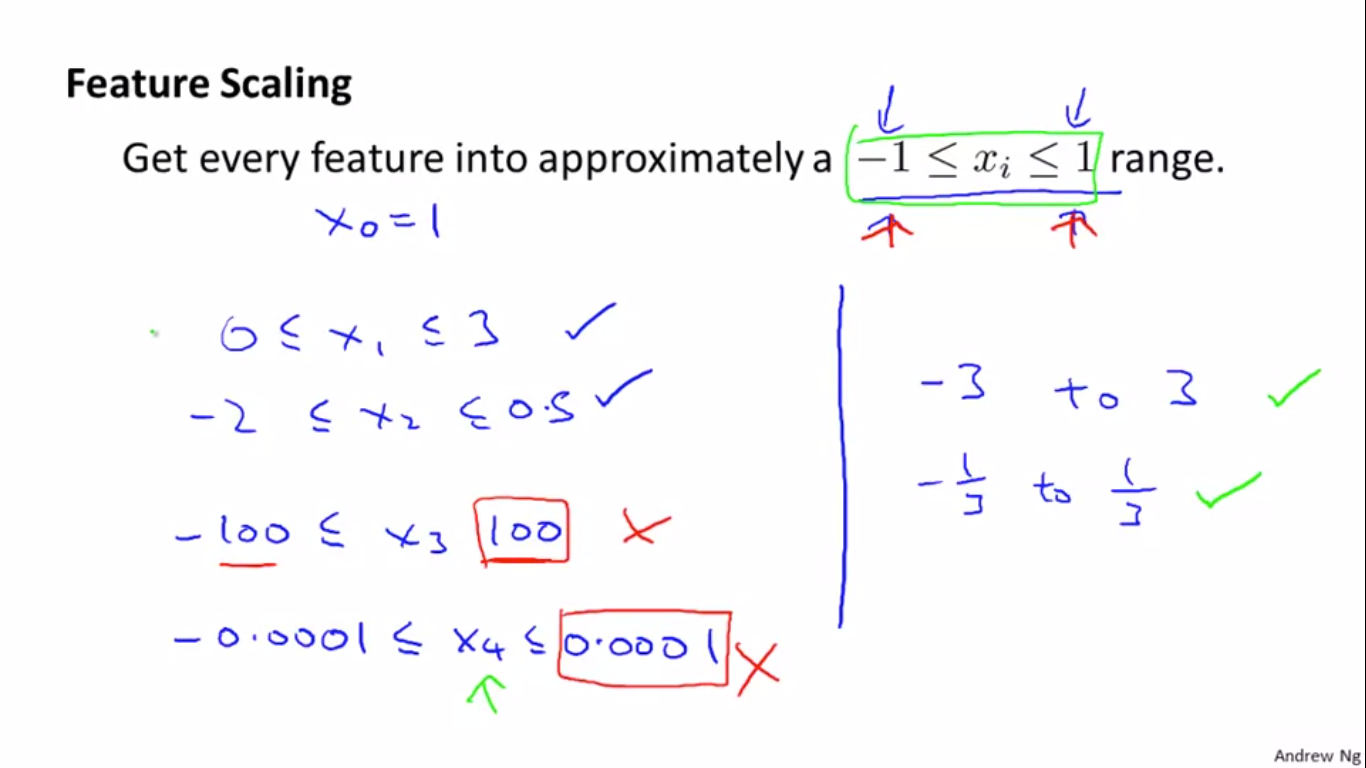
The way to prevent this is to modify the ranges of our input variables so that they are all roughly the same. Ideally:

−1 ≤ x (i)≤ 1 or −0.5 ≤ x (i)≤ 0.5 . (They are just preferred values and need not be exact).

These aren't exact requirements; we are only trying to speed things up. The goal is to get all input variables into roughly one of these ranges, give or take a few.

Two techniques to help with this are feature scaling and mean normalization.

Feature scaling involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just 1.



If the values are too far from the provided ranges then they have to be scaled .

Mean normalization involves subtracting the average value for an input variable from the values for that input variable resulting in a new average value for the input variable of just zero.

To implement both of these techniques, adjust your input values as shown in this formula:

xi := (x i − μ i) / s i

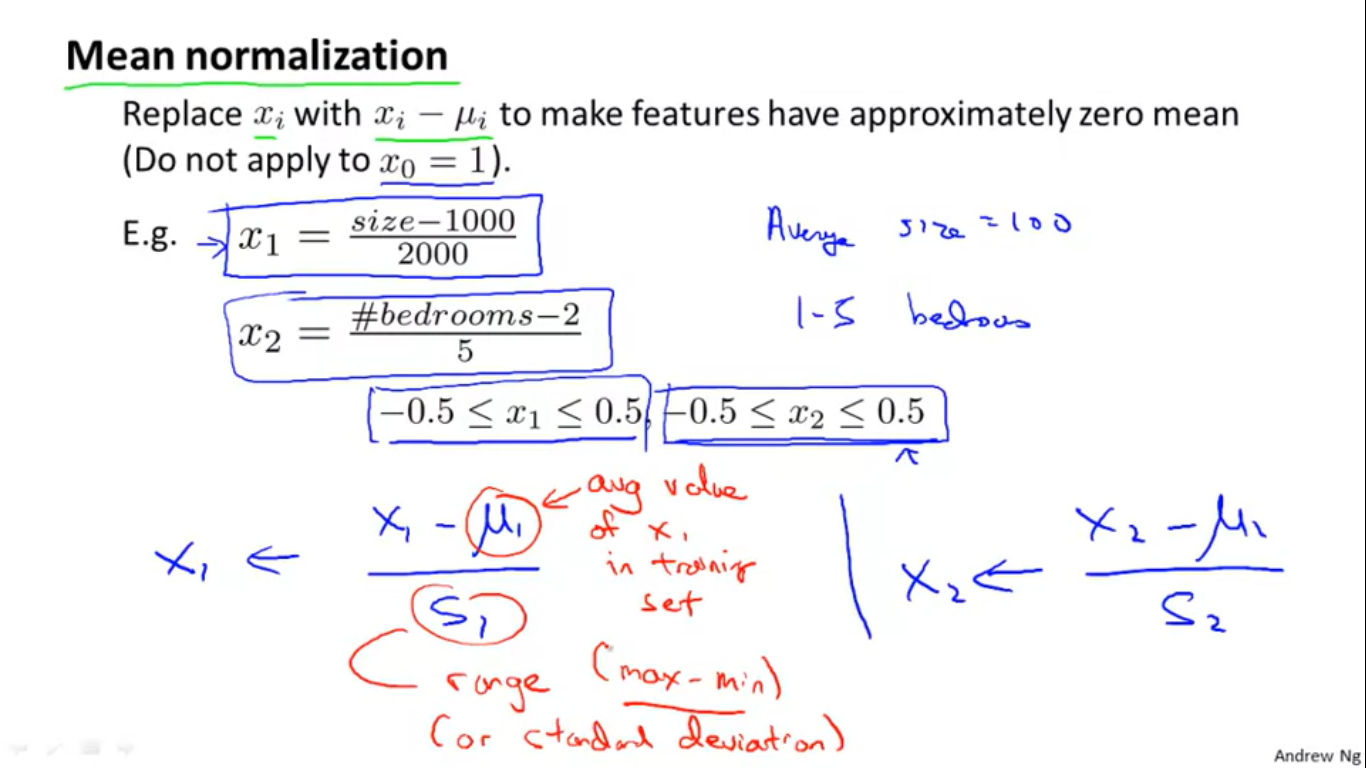
Where μ i is the average of all the values for feature (i) and

s i is the range of values (max - min), or s i is the standard deviation.

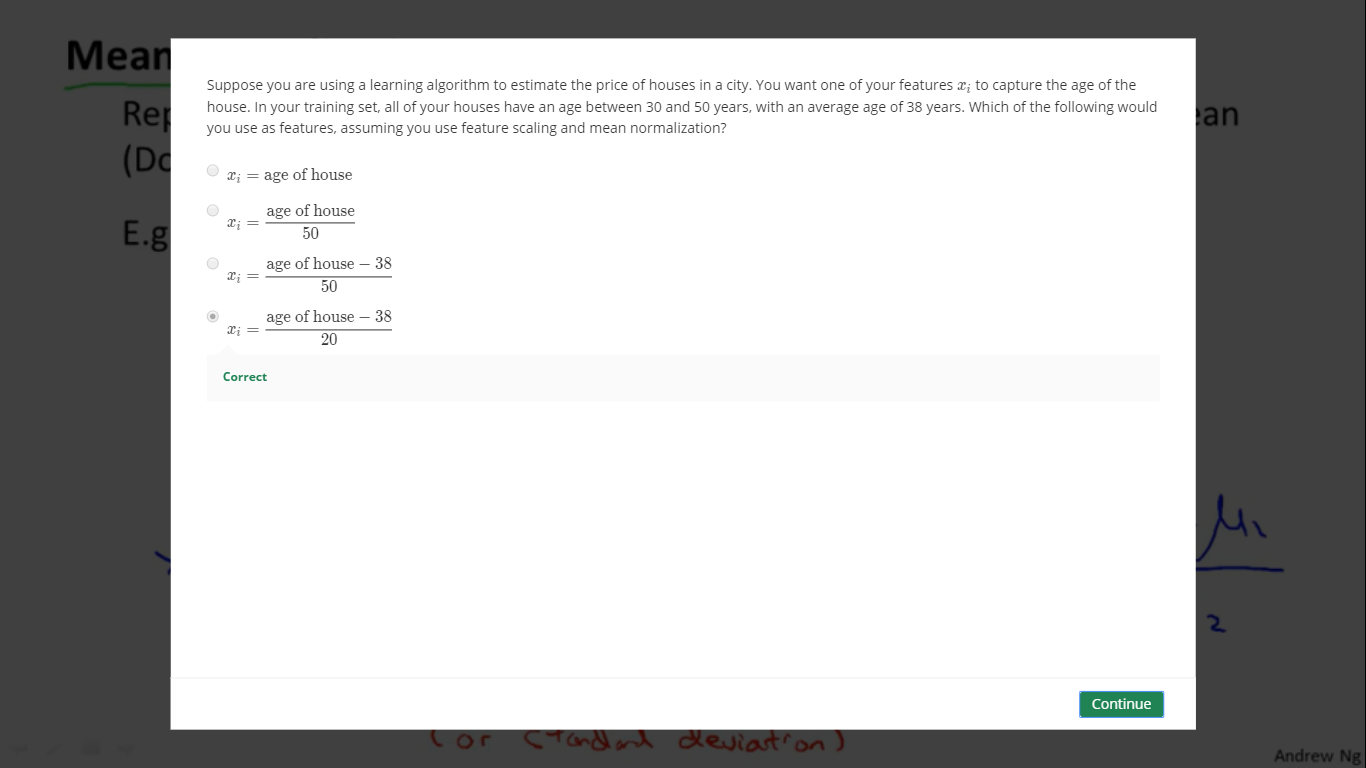
Note that dividing by the range, or dividing by the standard deviation, give different results. The quizzes in this course use range - the programming exercises use standard deviation.

For example, if x i represents housing prices with a range of 100 to 2000 and a mean value of 1000, then,

x\_i := {price-1000}/{1900}



Question :



Note : All the ratios are just the approximations and need not be the exact values .