**Multiple features**

Notation:

Multivariate linear regression:

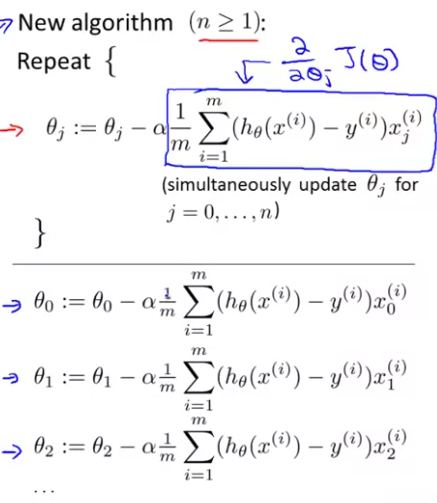
For a dataset consisting in n features, we can find out that the output value we want to predict, in function of the features multiplied by the parameters , can be expressed as:

If we define x0 = 1, we can say

**Gradient Descent for Multiple Variables**

Cost function:

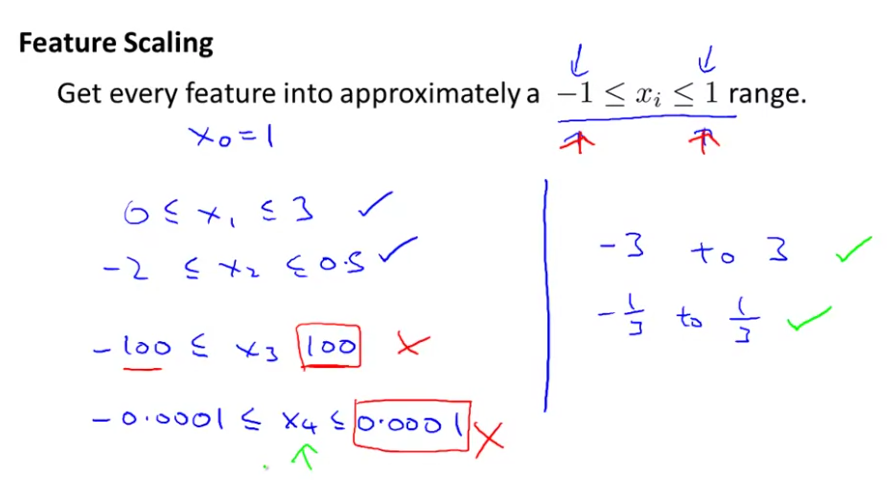
Gradient descent:

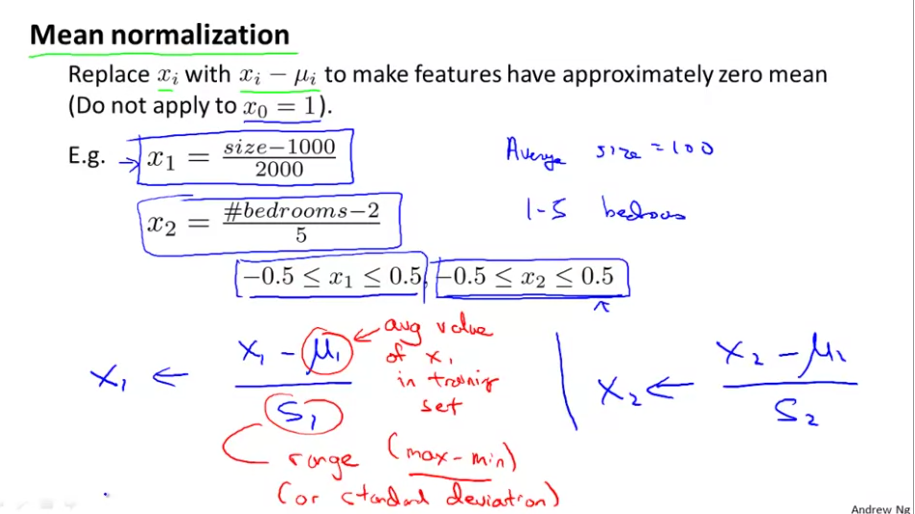


Learning rate

**Gradient Descent – Feature Scaling**

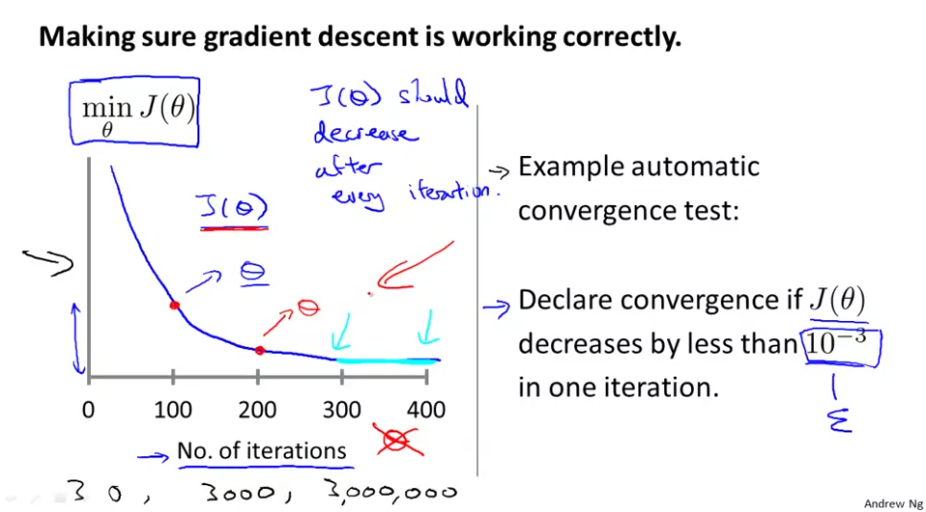
If you make sure that the different features are on a similar scale, you can make your gradient descent algorithm converge more quickly.



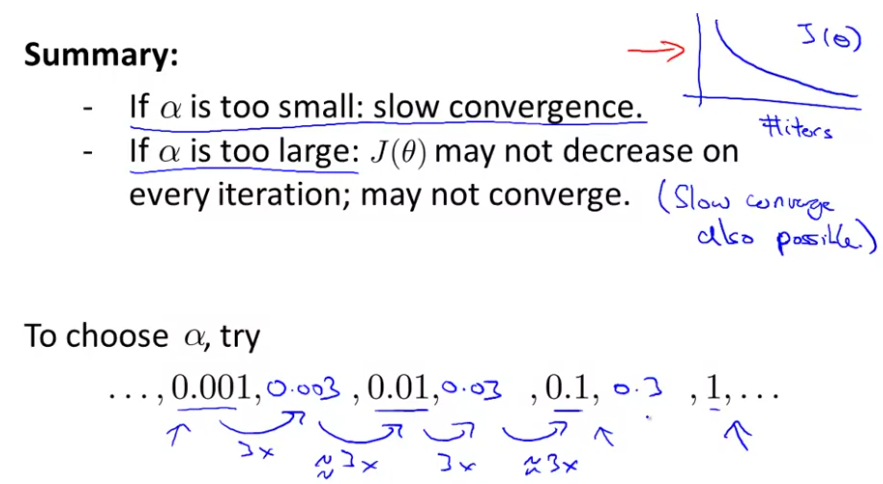


**Gradient Descent – Making sure gradient descent is working correctly and learning rate**

The gradient descent is working correctly if the function cost is descending after each iteration. So to check if the gradient descent is working fine, you can plot in function of the nº of iterations.



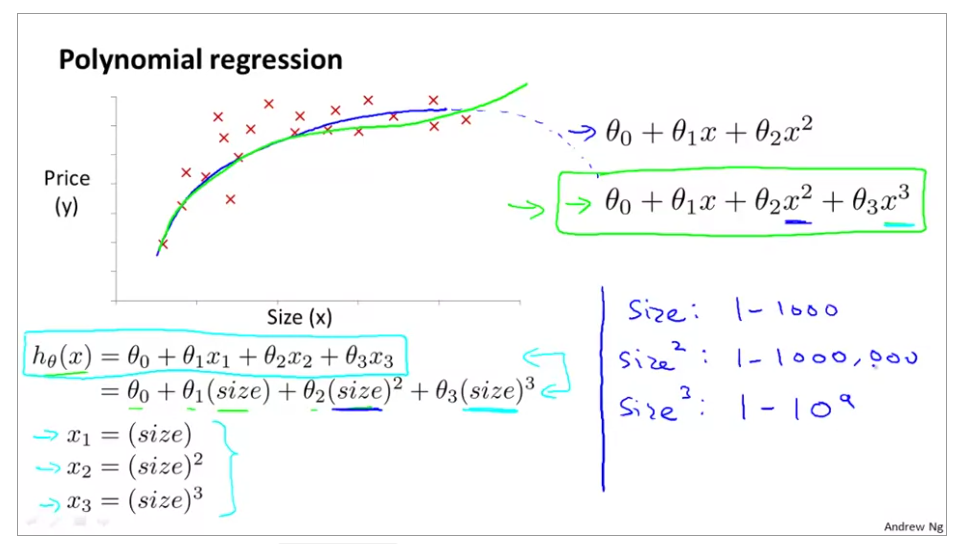
If you see that gradient descent isn’t converging, try using a smaller learning rate. However, if your learning rate is too small, gradient descent can be slow to converge. Existe un compromiso entre la velocidad de convergencia y la estabilidad/robustez del algoritmo.



**Features and polynomial regression**

You can transform your features in other features in order to obtain a better model. For example, you can use areas instead of the lengths, radius, etc.

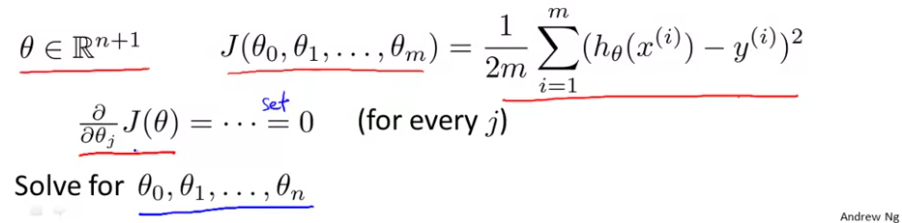
Sometimes linear regression just doesn´t fit well with the form of your database, so you can transform your function to suit better the data. An example:



Note that we are converting a 1-feature problem in a 3-feature problem.

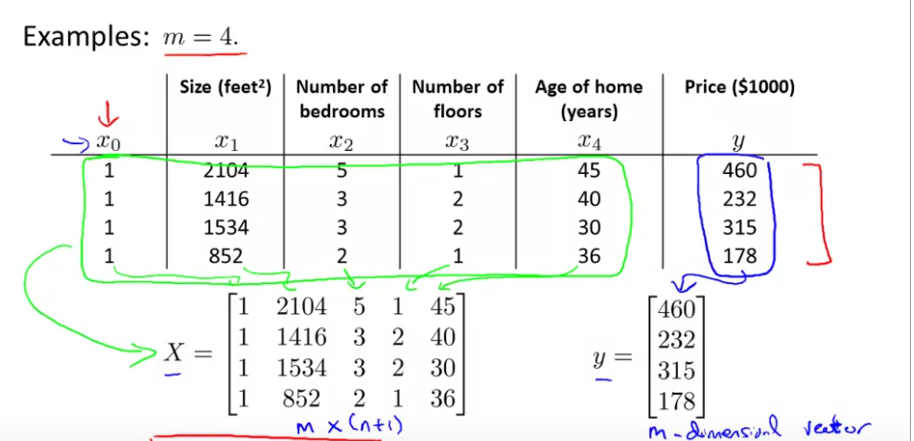
**Computing parameters analytically - Normal equation**

It lets you to calculate the parameters without having to iterate. It is based on the optimization principle, where you derivate the function and make it equal to 0:



If you are using matrices, this method is equivalent to:

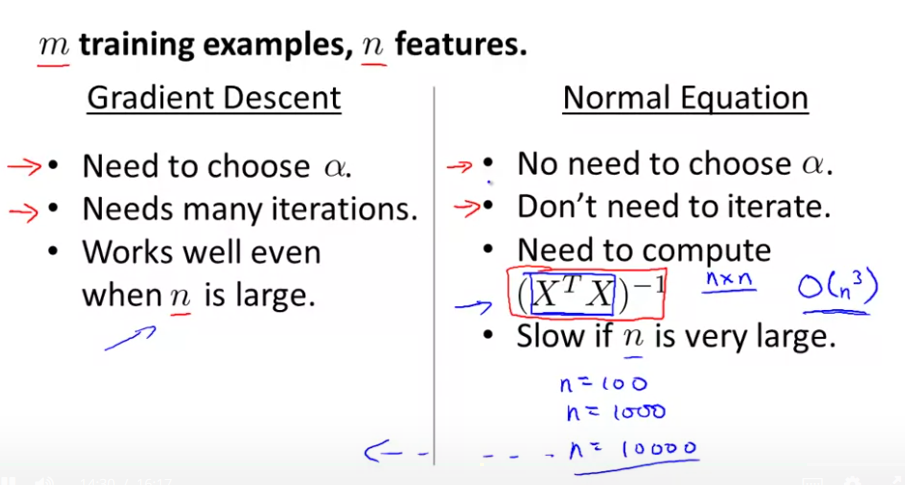
Example:



To do it in Matlab/Octave:



Note: With normal equation method, you don’t need to use feature scaling.



Good for problems with less than 10000 features

**Computing parameters analytically - Normal equation noninvertibility**

Not all matrices can be inverted, which is a necessary step for the normal equation method.

If XTX is non-invertible, it can be because there are redundant features (linearly dependent).

Another usual cause is because there are too many features (for example, if m is lower and n), but it can be solved if you use regularization or you just delete some features.