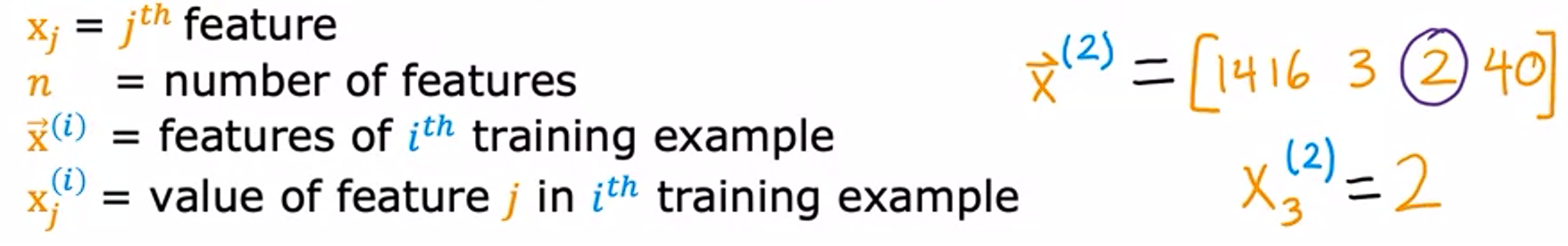
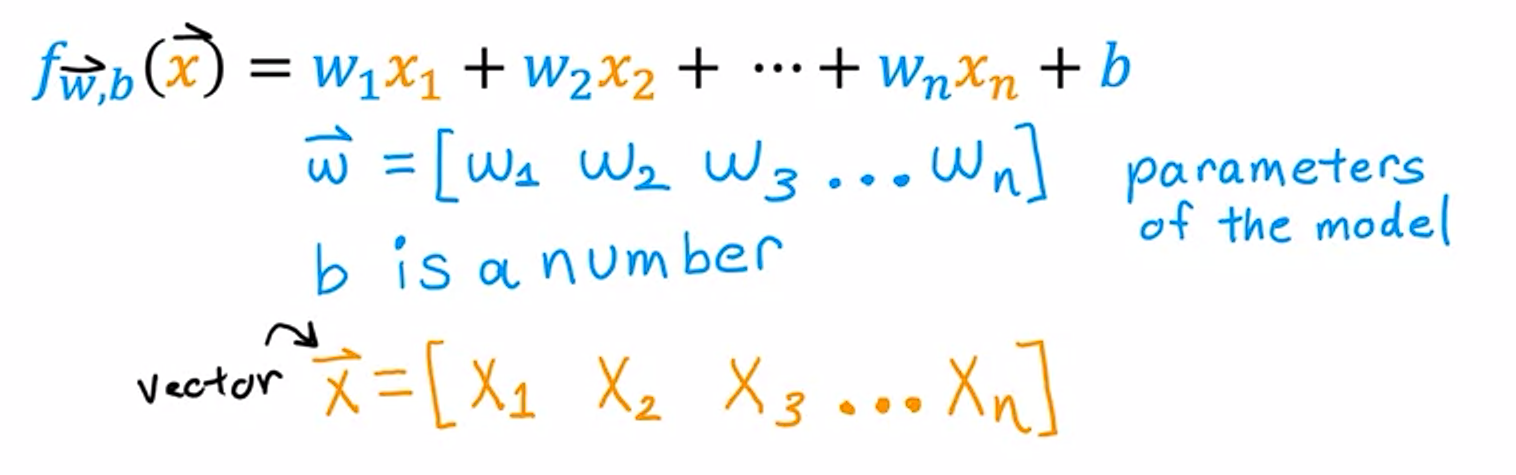
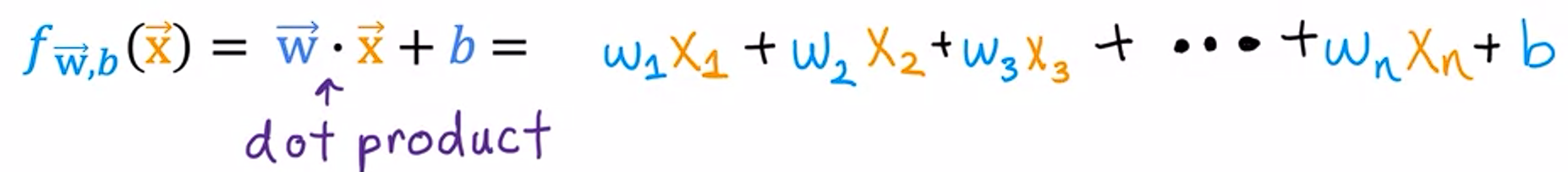
**Multiple Linear Regression**

There are n number of features (independent variables) instead of just one. (Each feature has m values, i.e., there are m training examples.) We use multiple independent variables (x1 x2 x3…) to predict the value of one dependent variable(y).

Multiple Linear Regression is also called Linear Regression With Multiple Features.







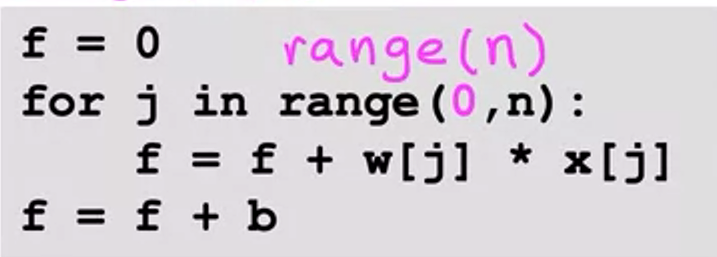
Note that Multiple Linear Regression, and Multivariate Regression ARE DIFFERENT.

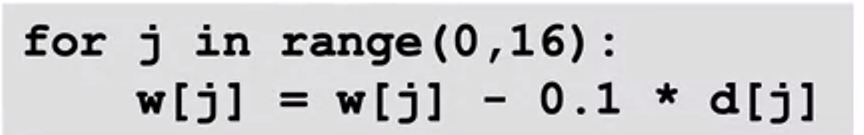
**Vectorization:**

Makes code shorter and more efficient. Writing vectorized codes helps us to use modern linear algebra libraries, and even use GPU. GPU can be used to execute code much more quickly when we write vectorized code.

NumPy dot function uses parallel hardware to efficiently calculate the dot product. So it is efficient and scales well for larger datasets.

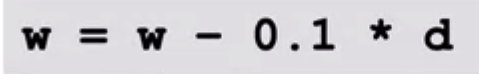
Without vectorization





With vectorization





**Normal Equation:**

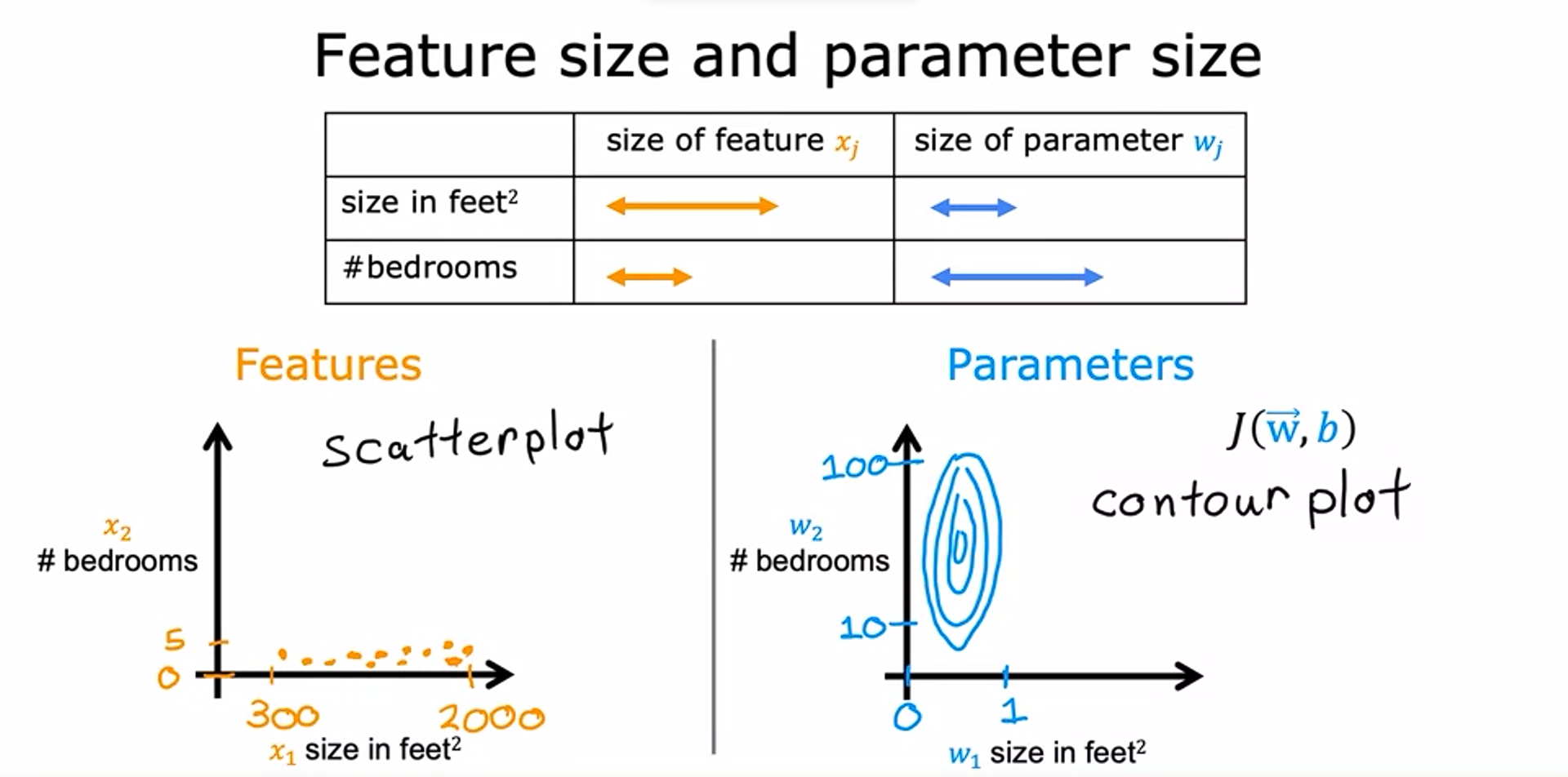
Alternative to Gradient Descent, only for linear regression. (unlike gradient descent, it is NOT generalized to other learning algorithms, such as the logistic regression algorithm)

It does not need iterative gradient descent. It uses an advanced linear algebra library to solve for w (vector) and b all in one go, without iterations.

It is quite slow if the number of features is large. (>10000)

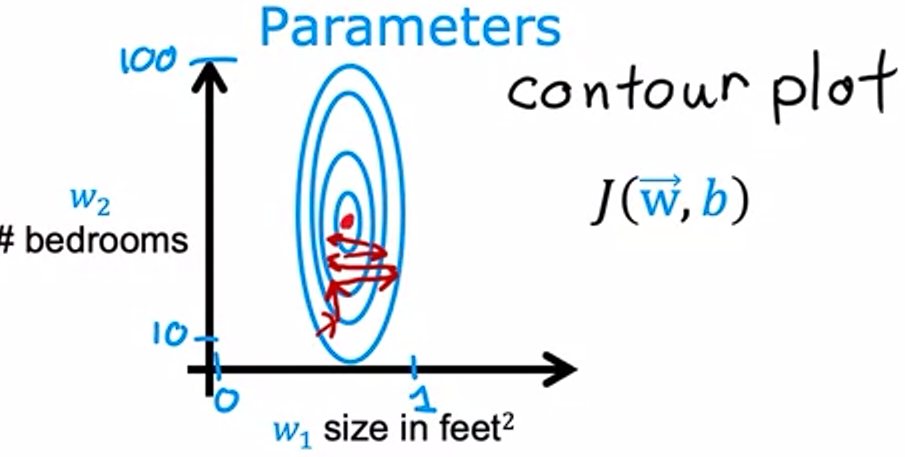
Normal equation may be used in machine learning libraries that implement linear regression.

It is a more complicated method.



When Size of one feature is larger than other, the scatter plot is skewed towards larger feature. (i.e., most of the points lie near the axis of size in feet2, as they range from 300 to 2000, while number of bedrooms range from just 0 to 5.)

The contour plot is opposite. The parameter for the larger feature (size in feet2) is being multiplied by a larger value already, so even small changes in this parameter are significant. So, parameter with features having larger values, tend to be smaller. Parameters of features with smaller values, need to be large to have a significant change in the cost. So here, on the contour plot, the range of values of parameter for number of bedrooms is larger, from 10 to 100, while parameter for size of feet2 is small, just between 0 and 1. So the contour plot is stretched along the smaller feature.



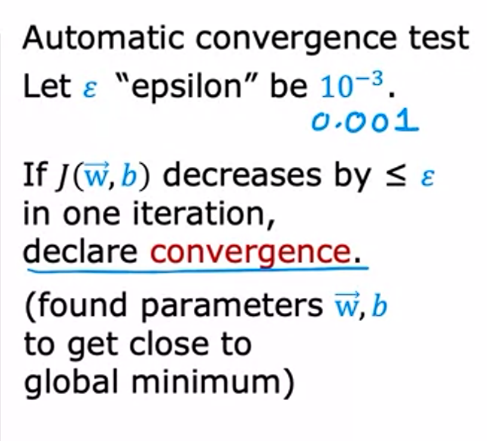
Because the contours are tall and skinny, gradient descent may end up bouncing back and forth for long time before reaching minimum.

Hence, we scale the features, so that all features range between 0 and 1.

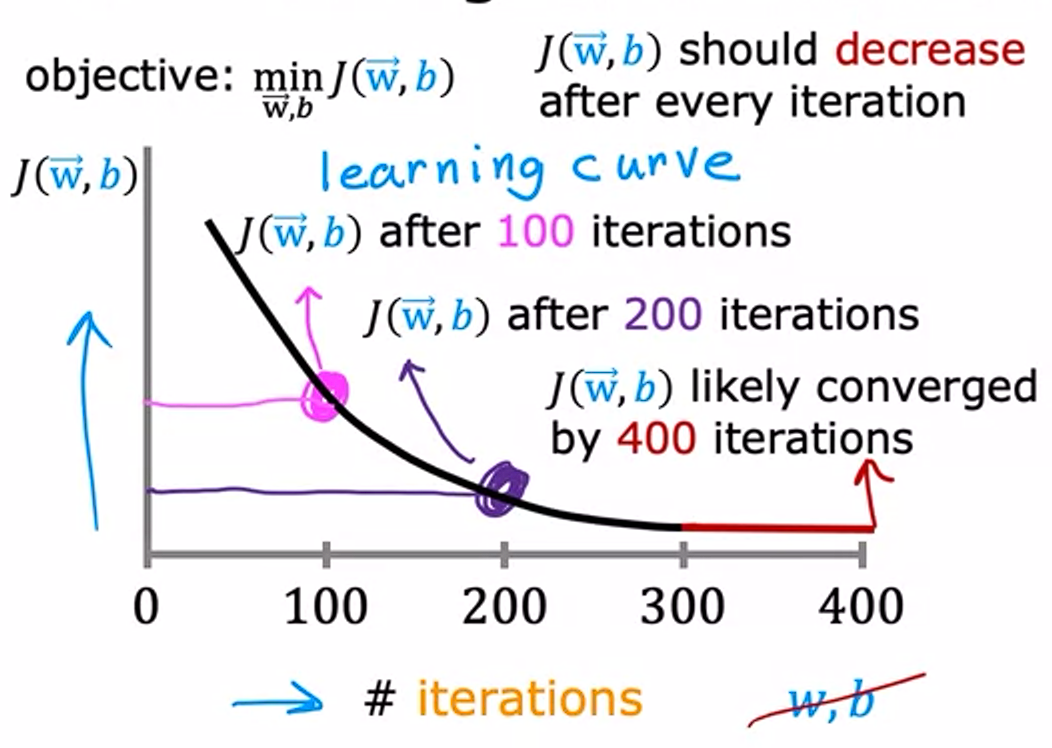
**Normalization:**

* Divide all values by the maximum value: x/max
* Mean Normalization: (x-mean)/(max-min)
* Z-Score Normalization: (x-mean)/(standard deviation)

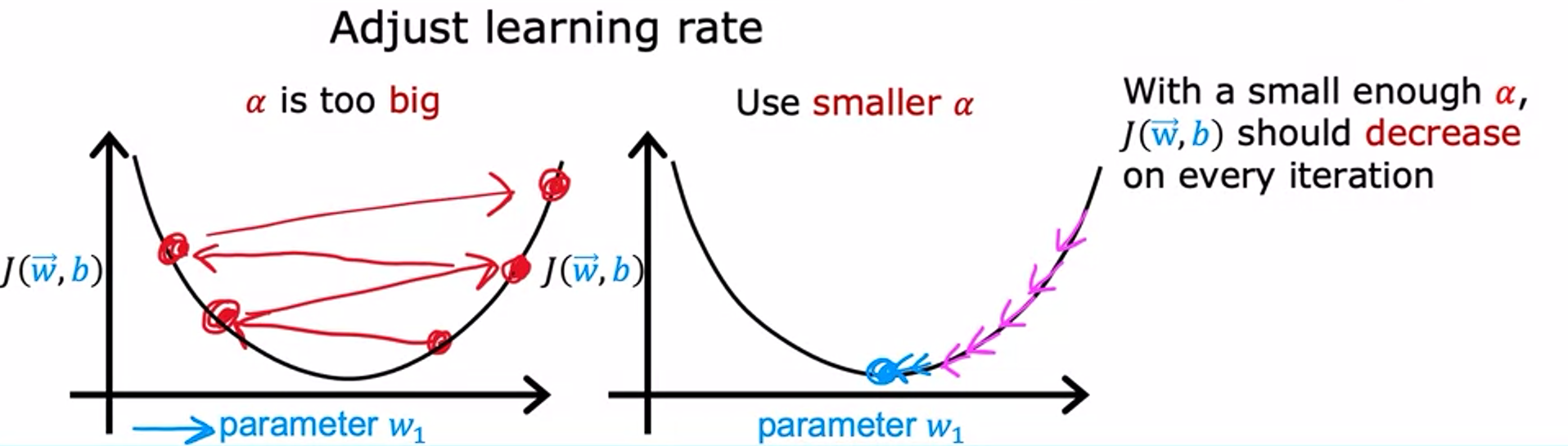
**Automatic Convergence Test:**



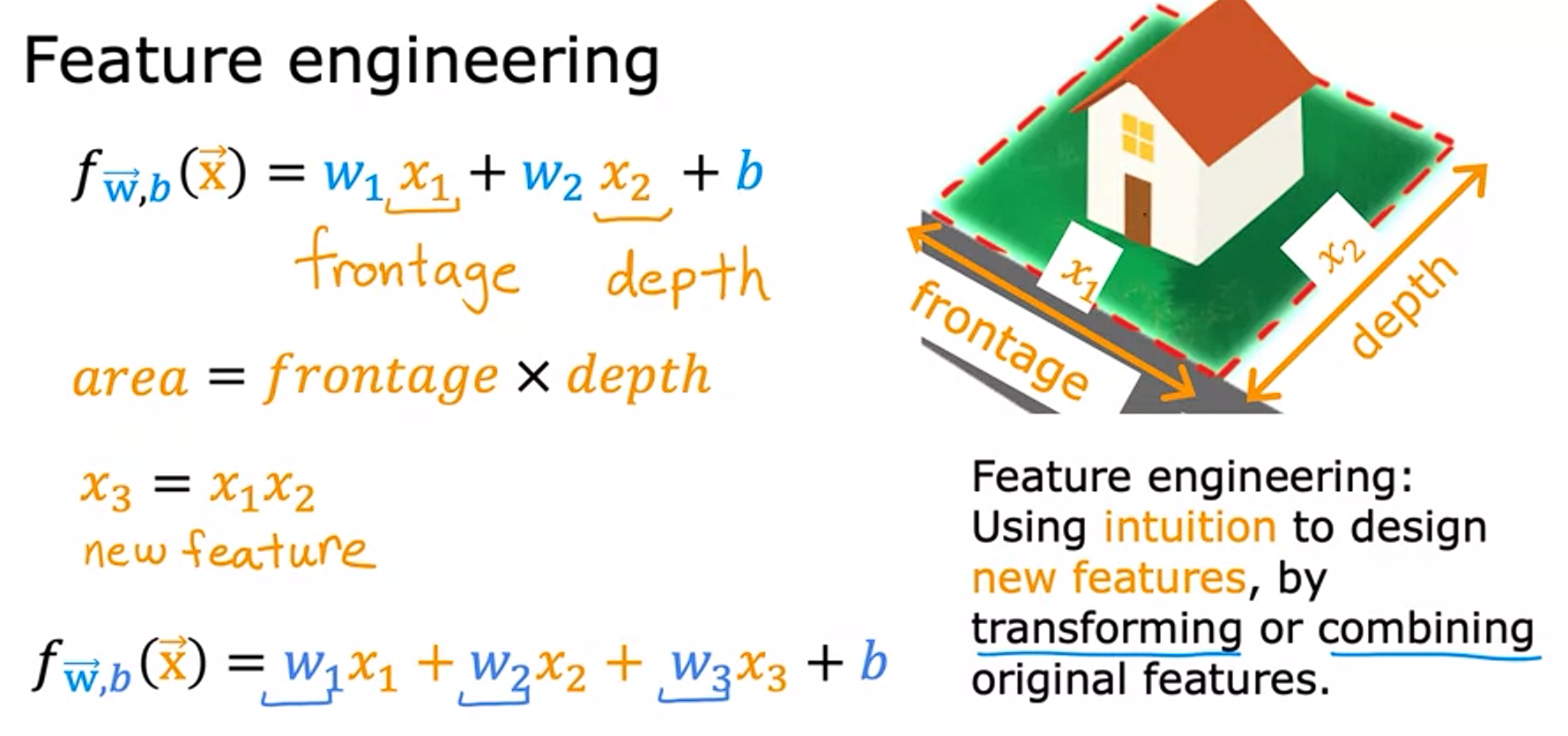
**Learning Curve:**



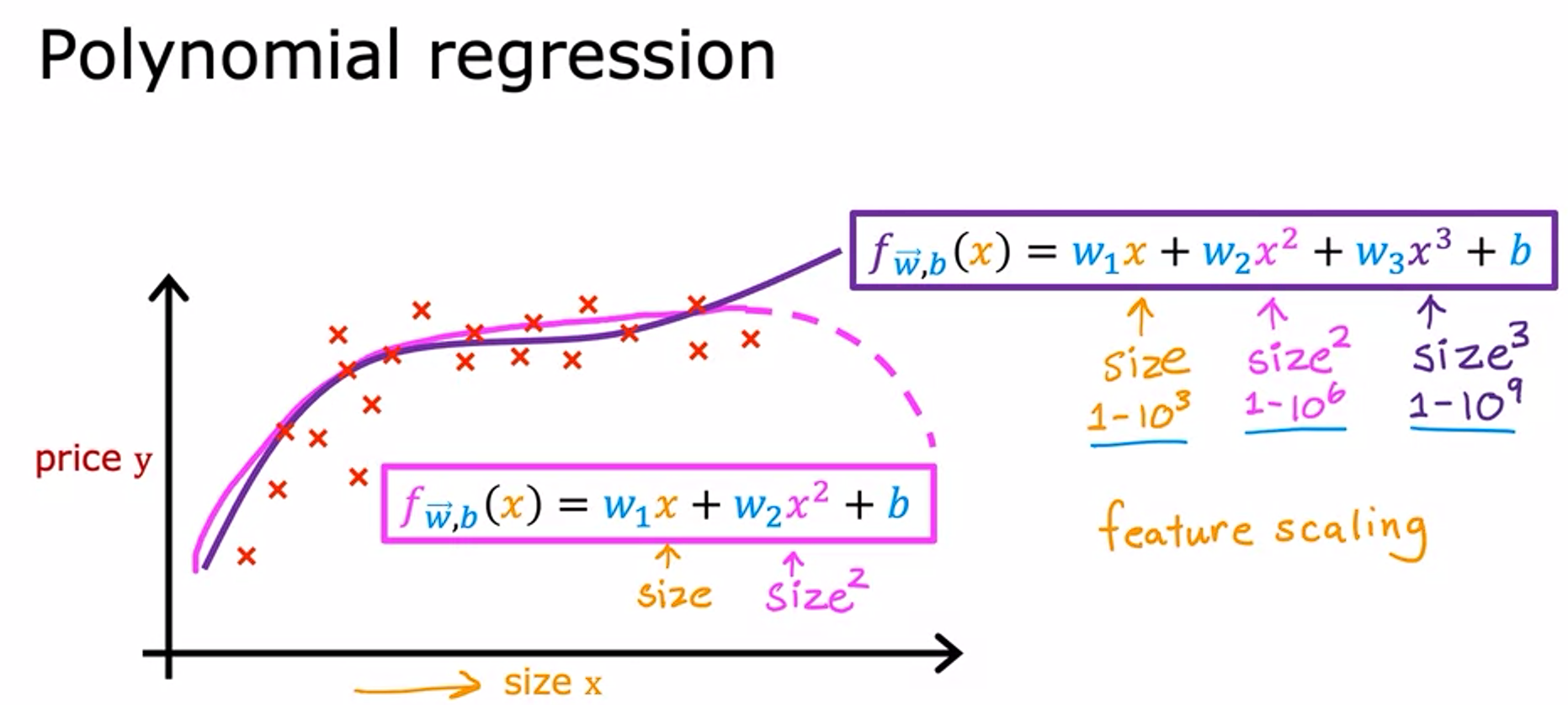
**Learning Rate:**



**Feature Engineering**



**Polynomial Regression**



When features are non-linear (like sqrt(x), or x3), or when they are combinations of other features (x1x2)

Feature Scaling (Normalization) becomes essential when using polynomial regression.

