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

x_j^i = value of feature j in the ith training example
$\mathbf{x}^i$ = the column vector of all the feature inputs of the ith training example
m = the number of training examples
n = $|\mathbf{x}^i|$; the number of features

The multivariable form of the hypothesis function accommodating these multiple features is as follows:

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

In general:

define $x_0 = 1$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \text{Then: } h_\theta(x) = \theta^T x$$

This is a vectorization of our hypothesis function.

The gradient descent equation itself is generally the same form; we just have to repeat it for our n features:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i$$

}

*Simultaneously update θ_j for $j = 0, \dots, n$)

Feature Scaling: make sure features are on a similar scale.
Get every feature into approximately a $-1 \le x_i \le 1$ range.

Mean normalization:
Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean.

To use both feature scaling and mean normalization:

$$x_i := \frac{x_i - \mu_i}{s_i}$$ where μ_i = average for all value for feature i, s_i = range of values (max-min), or the standard deviation

Polynomial Regression: The hypothesis function need not be linear if that does not fit the data well. We can change the behavior or curve of our hypothesis function by making it a quadratic, cubic, or square root function.

Ex. $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$ (create new feature $x_2 = x_1^2$)

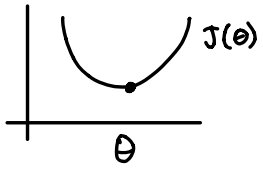
Normal Equation
Gradient descent gives one way of minimizing J. The Normal Equation performs the minimization explicitly and without resorting to an iterative algorithm. In the Normal Equation method, we will minimize J by explicitly taking its derivatives with respect to the θ_j's and setting them to zero. This allows us to find the optimum theta without iteration.

Intuition: If 1D ($\theta \in \mathbb{R}$)

$J(\theta) = a\theta^2 + b\theta + c$

$\frac{d}{d\theta} J(\theta) = \dots$ (set to 0)

Solve for θ



For $\theta \in \mathbb{R}^{n+1}$:

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

$$\frac{\partial}{\partial \theta_j} J(\theta) = \cdots = 0 \quad \text{(for every } j\text{)}$$

Solve for $\theta_0, \theta_1, \ldots, \theta_n$

For m examples $(x^1, y^1), \ldots, (x^m, y^m)$; n features

$$x^i = \begin{bmatrix} x_0^i \\ x_1^i \\ x_2^i \\ \vdots \\ x_n^i \end{bmatrix} \in \mathbb{R}^{n+1} \qquad X = \begin{bmatrix} — (x^1)^T — \\ — (x^2)^T — \\ \vdots \\ — (x^m)^T — \end{bmatrix} \quad \text{(design matrix)} \qquad y = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix}$$

The normal equation formula:

$$\theta = (X^T X)^{-1} X^T y$$

Gradient Descent
- Need to choose α
- Needs many iterations
- Works well even when n (features) is large

Normal Equation
- No need to choose α
- Don't need to iterate
- Need to compute $(X^T X)^{-1}$ $O(n^3)$
- Slow if n is very large