Multivariate Linear Regression.

Notation:

n: the number of features.

$x^{(i)}$: input features of ith training example, which is in n dimensions.

$x_j^{(i)}$: value of feature j in ith training example.

Hypothesis: $h_\theta(x) = \theta_0 + \sum_{i=1}^{n} x_i \cdot \theta_i, \quad x_0 = 1.$

Let $x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}, \quad \in R^{n+1}$

$h_\theta(x) = \theta^T \cdot x$.

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

Gradient descent:

Repeat until converge {

$$\theta_j = \theta_j - \alpha \cdot \frac{\partial}{\partial \theta_j} J(\theta)$$

$$= \theta_j - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}. \quad (\text{where } j = 0, \cdots, n)$$

}.

Feature scaling: make sure features are on a similar scale, get every feature into approximately a $-1 \leq x_i \leq 1$ range, so it can converge much faster.

Mean normalization: Replace x_i with $x_i - u_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).

E.g.: $x_1 = \frac{x_1 - u_1}{s_1}$ $s_1 \rightarrow$ range (max-min).

$J(\theta)$ should decrease after every iteration.

Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.

Use smaller α, if α is too large, $J(\theta)$ may not decrease on every iteration, may not converge.

Normal Equation: A New way to calculate θ.

m examples, $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})$;

n features.

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}, \quad \text{then design matrix } X = \underset{m \times (n+1)}{\begin{bmatrix} \cdots (x^{(1)})^T \cdots \\ (x^{(2)})^T \\ \vdots \\ \cdots (x^{(m)})^T \cdots \end{bmatrix}}$$

$$Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}.$$

$$\theta = (X^T \cdot X)^{-1} \cdot X^T \cdot Y,$$

$(X^T \cdot X)^{-1}$ is inverse of matrix. $X^T \cdot X$.

Gradient Descent.	Normal Equation.
i) Need to choose α.	i). No need to choose α.
ii). Needs many iterations.	ii). Don't need to iterate.
iii). Work well even when n is large.	iii). Need to compute $(X^T \cdot X)^{-1}$. ... $O(n^3)$
	iv) slow if n is very large, but fast when n is not so large.

Normal Equation: $\theta = (X^T \cdot X)^{-1} \cdot X^T \cdot y$.

What if $X^T \cdot X$ is non-invertable? (singular / degenerate).

i). Redundant features. (features are linearly dependent).

ii). Too many features: delete some features, or use regularization.