Multiple Linear Regression

—— Algorithm implemented in Octave
—— Simple data were applied with the algorithm

—— Same data were analyzed in R
—— Same data were analyzed in Python

Algorithm

$x_0 = 1$

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\cancel{\theta_0, \theta_1, \ldots, \theta_n}$ θ n+1 - dimensional vector

Cost function:

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

$J(\theta)$

Repeat {

$\frac{\partial}{\partial \theta_j} J(\theta)$

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

(simultaneously update θ_j for $j = 0, \ldots, n$)

$x_0^{(i)} = 1$

}

Gradient descent in practice I: Feature Scaling

Idea: Make sure features are on a similar scale.

Gradient descent in practice II: Learning rate

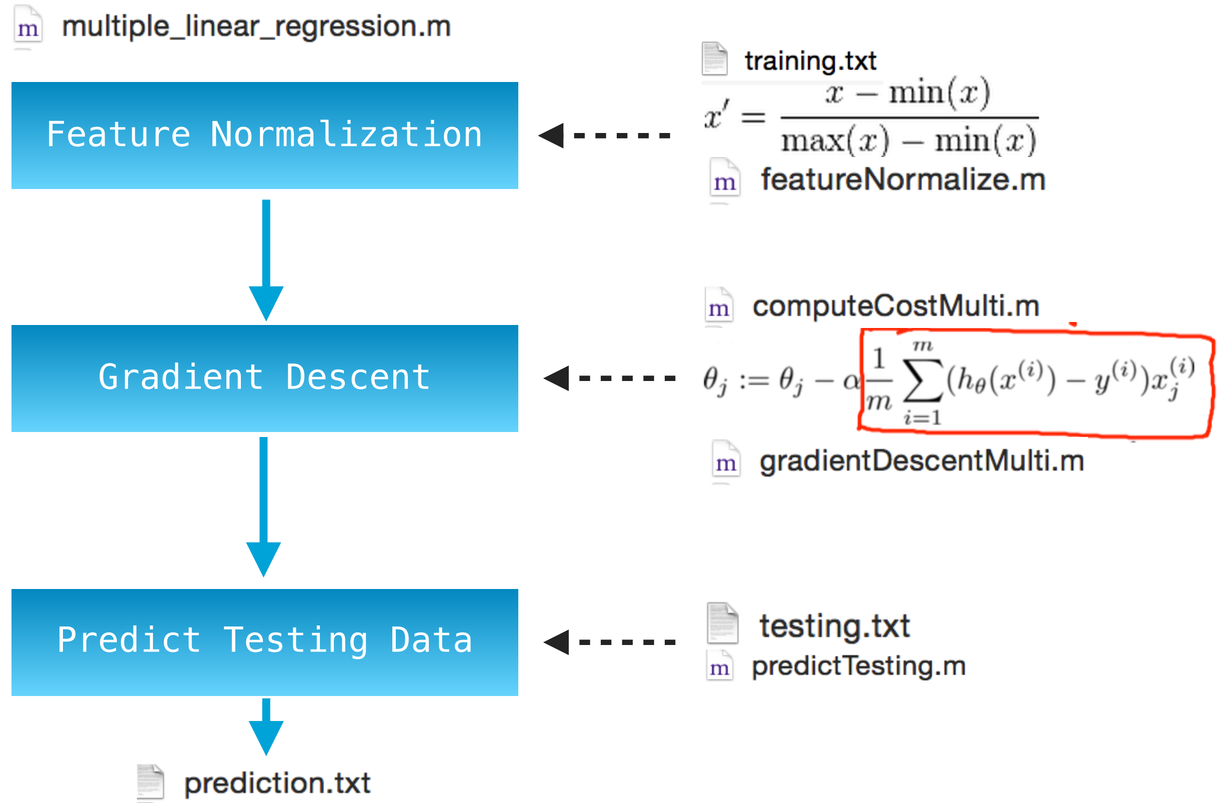
- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge. (Slow

To choose α, try

$\ldots, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \ldots$

3x ≈3x 3x ≈3x

Implemented in Octave



Data Analyzed in R

File Name:
multiple_linear_regression.R

Usage:
/usr/bin/Rscript multiple_linear_regression.R training.txt testing.txt prediction.txt

Core Functions{Packages} Used:
build model
lm{stats}
make prediction on testing data
predict{stats}

Data Analyzed in Python

File Name:
multiple_linear_regression.py

Usage:
python multiple_linear_regression.py training.txt testing.txt prediction.txt

Core Functions{Modules} Used:
build model
linear_model.LinearRegression().fit(){sklearn}
make prediction on testing data
linear_model.LinearRegression().predict(){sklearn}