

# Multivariate Linear Regression

Sidharth Baskaran

June 2021

## Multiple Features

- Notation usage
  - $n$  - number of features
  - $m$  - number of training examples
  - $x^{(i)}$  - input features of  $i$ th training example -> is a vector with dimension  $n$
  - $x_j^{(i)}$  - number of feature  $j$  in  $i$ th training example
- Hypothesis is of the form  $h_\theta(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$ 
  - Define  $x_0 := 1$  as coefficient of  $\theta_0$ 
    - \* Let  $x = \begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}$  and  $\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix}$
    - \* Both are in  $\mathbb{R}^{n+1}$
  - Thus  $h_\theta(x) = \theta^T x$  where  $\theta$  is transposed to allow multiplication with matrix  $x$

## Gradient descent with multiple features

- Let  $\theta$  be the  $n + 1$ -dimensional parameter vector
- Perform  $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$ 
  - Simultaneous update of  $\theta_j$  for  $j = 0, \dots, n$

repeat until convergence:

$$\begin{aligned}\theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)} \\ \theta_1 &:= \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)} \\ \theta_2 &:= \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}\end{aligned}$$

## Feature Scaling

- Make sure features are on similar scale or interval
  - Contours will become more like circles -> gradient descent is less complicated
- Get feature into approx.  $-1 \leq x_i \leq 1$  range
- Mean normalization
  - Replace  $x_i$  with  $x_i - \mu_i$  to make features have an approximately zero mean (except  $x_0 = 1$ )
- Thus formula for feature scaling with mean normalization is  $x_i \rightarrow \frac{x_i - \mu_i}{s_i}$  where  $s_i$  can be range of values in training set or the std. dev

## Learning rate

- Debugging gradient descent

- Plot  $\min J(\theta)$  as gradient descent runs over # of iterations
  - Should show decrease as iterations progress
- Automatic convergence test
  - Ex: Declare convergence if  $J(\theta)$  decreases by less than  $10^{-3}$  in 1 iteration
- For sufficient small  $\alpha$   $J(\theta)$  decreases on every iteration
  - If too small -> slow to converge

## Features and Polynomial Regression

- Can define a feature in terms of others -> combining features
- Fitting polynomial model
  - Define each feature to be square, cubed, etc.
  - Apply regular linear Regression
  - Feature scaling is important -> exponential values increase scale

## Normal Equation

- Method to solve for  $\theta_0, \dots, \theta_n$  through derivative of  $J$  with respect to  $\theta_j$ , set to 0, and minimization
- Construct a design matrix  $X$  that is of dimension  $m \times (n + 1)$  and contains all of the training data
  - Each feature vector  $x^{(i)} \in \mathbb{R}^{n+1}$  is transposed to constitute a row of  $X$
- Construct vector  $y$  which contains the result/expected output and is  $m$ -dimensional
- Optimum  $\theta$  is given by  $\boxed{(X^T X)^{-1} X^T y}$
- Feature scaling not required
- Gradient descent comparison
  - Gradient descent -> need to choose  $\alpha$  and needs many iterations
    - \* Works well for a large  $n$
  - Normal equation -> no need to choose  $\alpha$  and don't need to iterate
    - \* Slow if  $n$  is very large -> need to compute  $(X^T X)^{-1}$ 
      - Inversion is  $O(n^3)$
- $X^T X$  can be noninvertible
  - `pinv` works regardless of invertibility and `inv` will throw error in octave
  - Redundant features (i.e. linear dependence)
  - Too many features means  $m \leq n$