Applied
Machine Learning

Lecture: 3
Multiple Linear Regression

Ekarat Rattagan, Ph.D.

Outline

3.1 Multiple features

Single features (variable)

Size (feet²) x	Price ($1000) y
2104	460
1416	232
1534	315
852	178
…	…

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Multiple features (variables)

Size (feet^2)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example.

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

Hypothesis

Single variable $h_\theta(x) = \theta_0 + \theta_1 x$

Multiple variables $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Multivariate linear regression

$$h_\theta(x) \quad = \quad \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n \quad = \quad \theta^T x$$

$$\theta^T x = \left[\theta_0, \theta_1, \quad \ldots \quad, \theta_n\right] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

For mathematical convenient, we define $x_0^{(i)} = 1$

3.2 Gradient descent for multiple variables

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

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

Gradient descent:

Repeat until convergence $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

Simultaneously update for every $j = 0, \ldots, n$

$\}$

Gradient Descent for single variable

Single variable, i.e., $j = 0$ and 1

Repeat until convergence $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$\}$

Gradient Descent for multiple variables

Multiple variables, i.e., j = 0, 1, …, n

Repeat until convergence $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

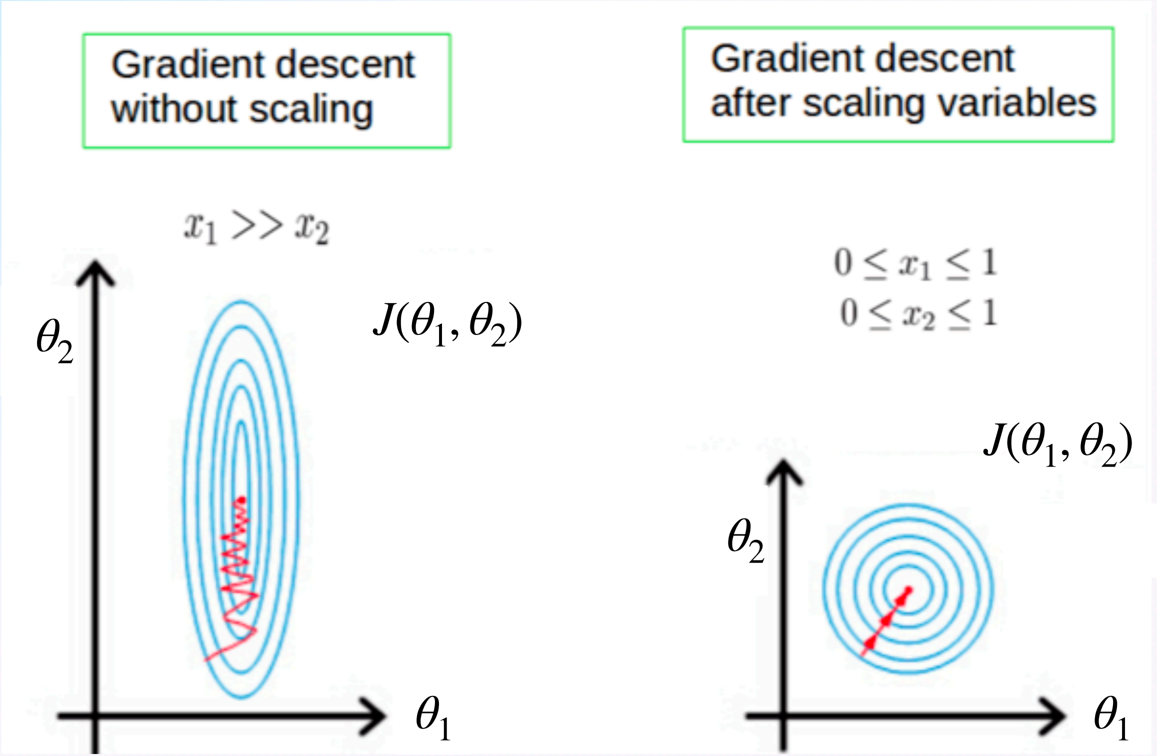
$$\vdots$$

$$\theta_n := \theta_n - \alpha \frac{1}{m} \sum (h_\theta(x^{(i)}) - y^{(i)}) x_n^{(i)}$$

$\}$

3.3 Feature Scaling

Feature Scaling



Source: https://stackoverflow.com/questions/46686924/why-scaling-data-is-very-important-in-neural-networklstm/46688787#46688787

Feature scaling

The goal is to transform features to be on a similar scale. This improves the performance and training stability of the model.

Standardization
(or Z-score normalization)

$$x_i' = \frac{x_i - \mu}{s}$$

Min-max scaling
(or normalization)

$$x_i' = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

https://sebastianraschka.com/Articles/2014_about_feature_scaling.html

https://developers.google.com/machine-learning/data-prep/transform/normalization

3.4 Stochastic Gradient Descent

Stochastic Gradient Descent

Batch gradient descent $$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad (2)$$

2.2 Stochastic gradient descent

The *stochastic gradient descent* (SGD) algorithm is a drastic simplification. Instead of computing the gradient of all ***m*** examples, each iteration estimates this gradient on the basis of a single randomly picked example $x^{(i)}$:

$$\theta_j := \theta_j - \alpha (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad (4)$$

The stochastic process $\{\theta_j, \ \ j = 1,2,3,...\}$ depends on the examples randomly picked at each iteration. It is hoped that (4) behaves like its expectation (2) despite the noise introduced by this simplified procedure.

Ref: Bottou, L. (2012). Stochastic gradient descent tricks. In Neural networks: Tricks of the trade (pp. 421-436). Springer, Berlin, Heidelberg.

Batch Gradient Descent

repeat until convergence $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

$\}$

Stochastic Gradient Descent

1. Randomly shuffle (reorder) training examples

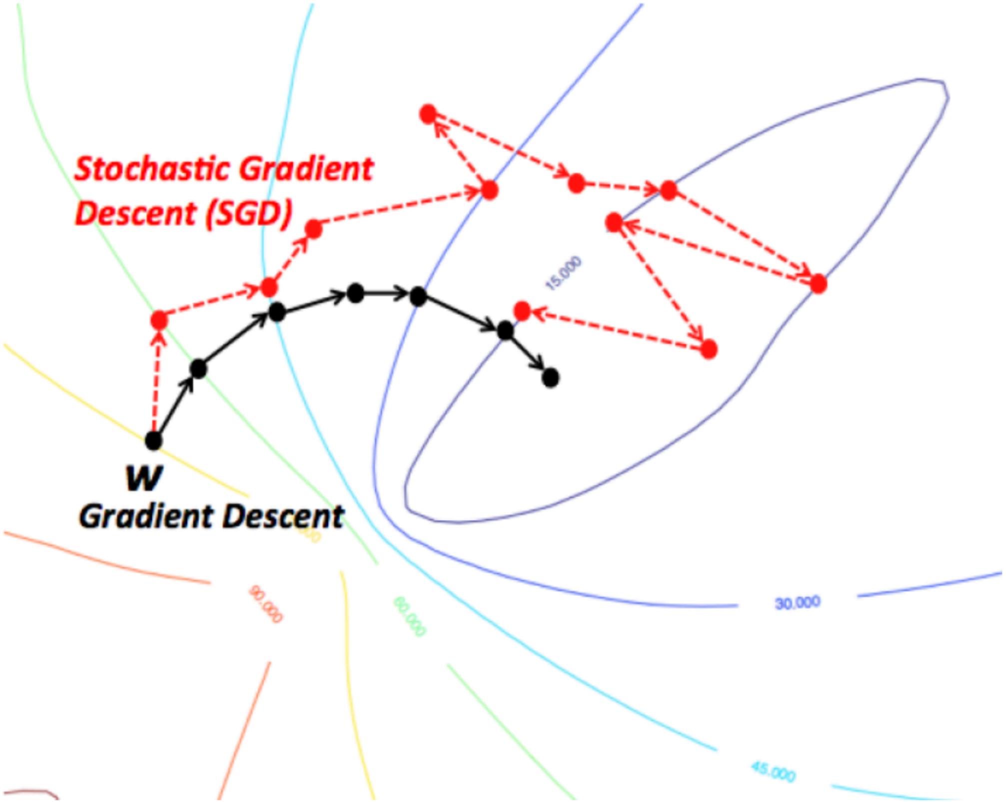
2. Repeat {
   for $i := 1, \ldots, m${

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

   (for every $j = 0, \ldots, n$)

   }

   }



picture source : https://wikidocs.net/3413

Learning schedule

- The solution to reduce the stochastic noise
- To gradually reduce the learning rate

Use learning rates of the form $\alpha_t := \dfrac{\alpha_0}{(1 + \alpha_0 \lambda t)}$

Ref: Bottou, L. (2012). Stochastic gradient descent tricks. In Neural networks: Tricks of the trade (pp. 421-436). Springer, Berlin, Heidelberg.

3.5 Mini-batch Gradient Descent

Mini-batch gradient descent

Batch gradient descent: Use all m examples in each iteration

Stochastic gradient descent: Use 1 example in each iteration

Mini-batch gradient descent: Use b examples in each iteration

Credit: Andrew NG

Mini-batch gradient descent

Say $b = 10, m = 1000$.

Repeat {

for $i = 1, 11, 21, 31, \ldots, 991$ {

$$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$$

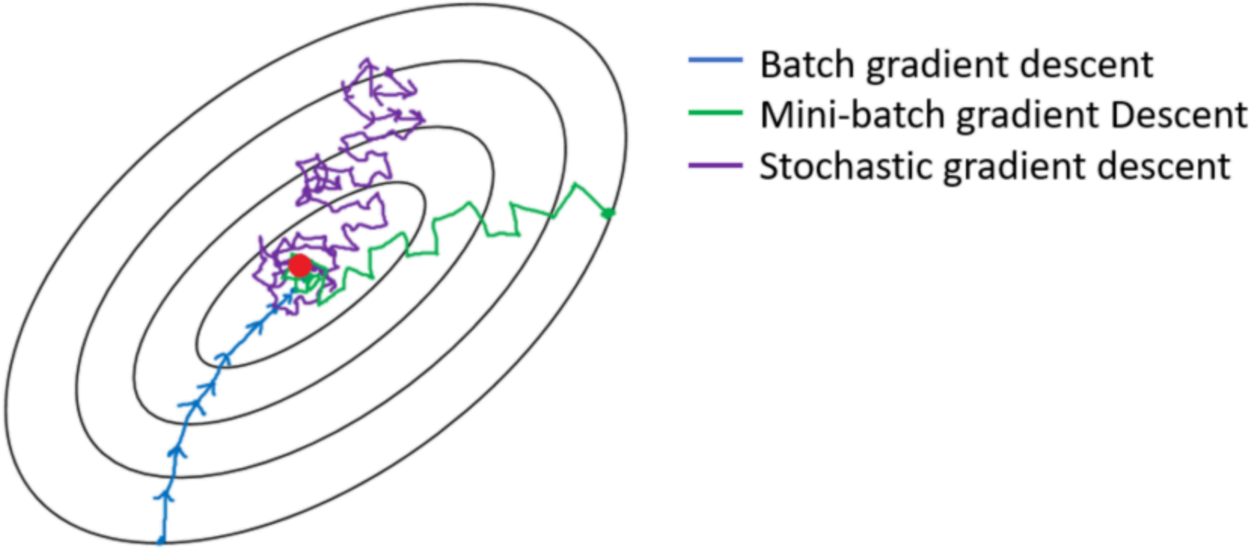
(for every $j = 0, \ldots, n$)

}

}

Credit: Andrew NG

Variants of Gradient Descent



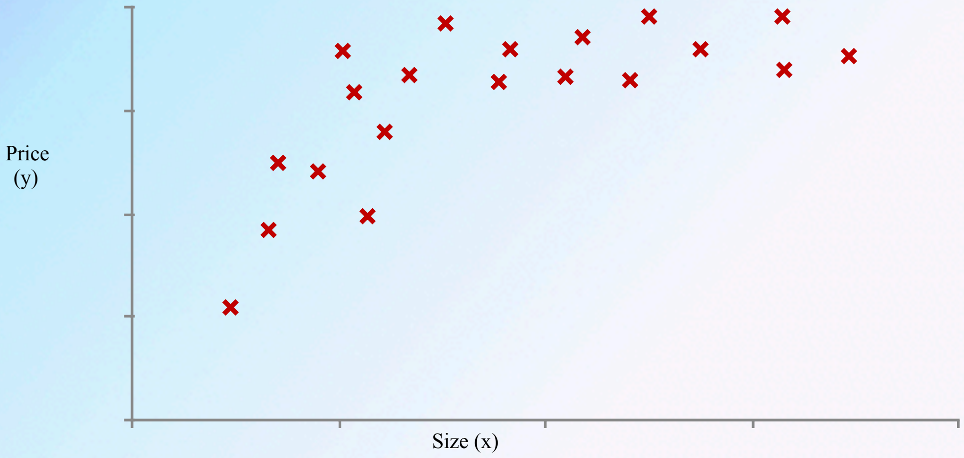
Credit: Andrew NG

3.6 Polynomial regression

Housing prices prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times front + \theta_2 \times depth$$

Polynomial regression



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3 \end{aligned}$$

$$\begin{aligned} x_1 &= (size) \\ x_2 &= (size)^2 \\ x_3 &= (size)^3 \end{aligned}$$