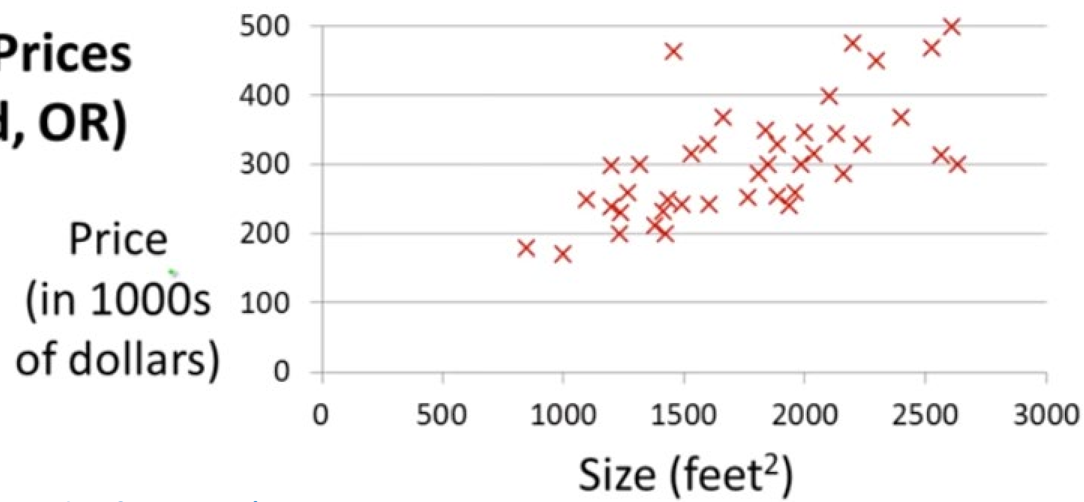
Multivariate Linear Regression

Housing Prices (Portland, OR)



(Single feature)

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

(Single variate linear regression)

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

Predicting house price y based on single feature of house size x

Multiple features (variables).

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Notataion

n: number of features

x^i: features of *i*-th training sample

$x_j{}^i$: value of feature *j* in *i*-th training sample

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n$$

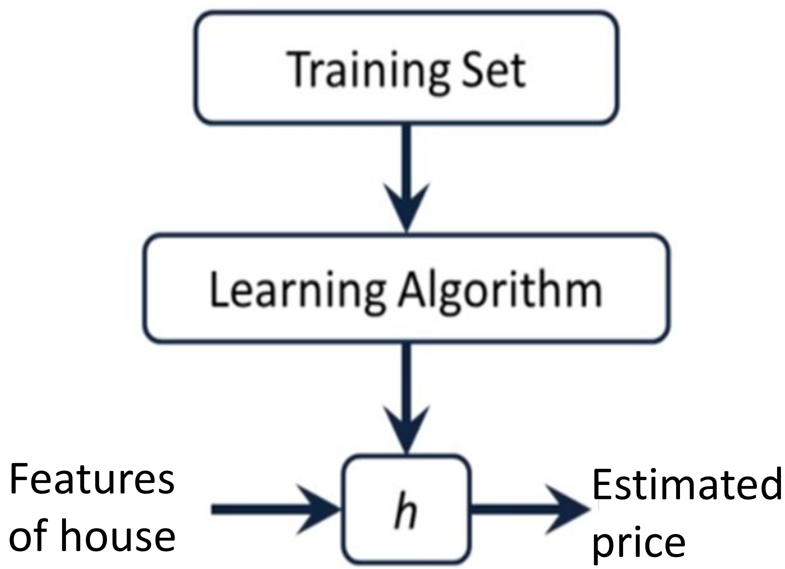
$$x = \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix} \in R^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} \in R^{n+1}$$

θ_0: bias, or consider x_0=1

In matrix form,

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

$\longrightarrow$ Multivariate linear regression



Hypothesis /
predictive model

To represent *h* – need a structure/model & model parameters

Multivariate linear regression:

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

x: features of house

θ: parameter of model

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

Parameters $\theta_0, \theta_1, \ldots, \theta_n$

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

Update parameters θ_0 θ_1 … θ_n using gradient descent until convergence

Simultaneously update θ_j , $j = 0, 1, \ldots, n$, according to

$$\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \ldots, \theta_n)$$

η is a positive number, which is called the learning rate
– too small, slow learning; too large, divergence

Prepare Data

To make sure features are on a similar scale, e.g., within (-1, 1) range

Feature scaling

E.g. x_1 = size (0-2000 feet^2)

x_2 = number of bedrooms (1-5)

$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$

Mean normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).

E.g. $x_1 = \frac{size - 1000}{2000}$

$x_2 = \frac{\#bedrooms - 2}{5}$

$$-0.5 \le x_1 \le 0.5, -0.5 \le x_2 \le 0.5$$

Use z score:

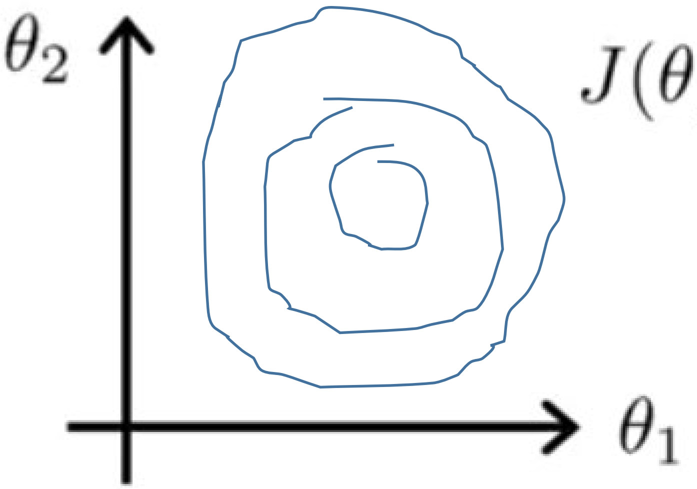
$$z = \frac{x - \mu}{\sigma}$$

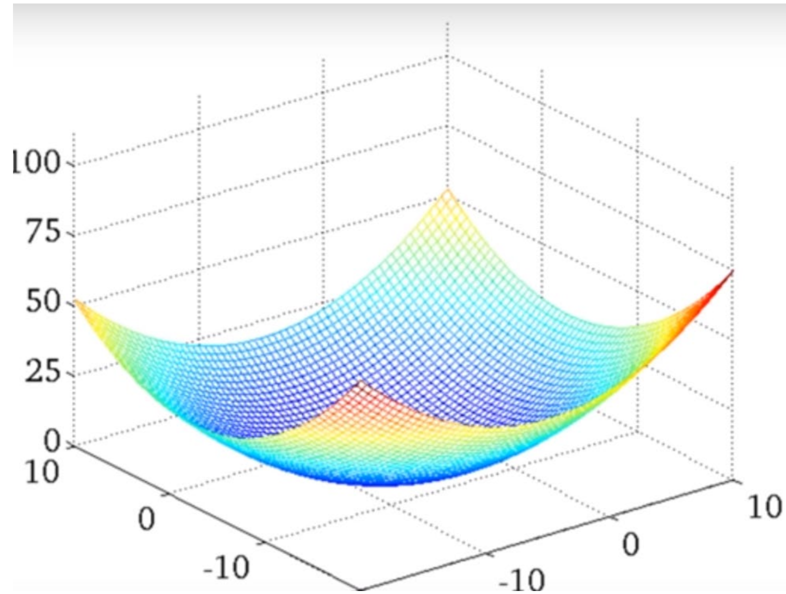
where μ is the mean of feature x, and σ is the standard deviation of x.

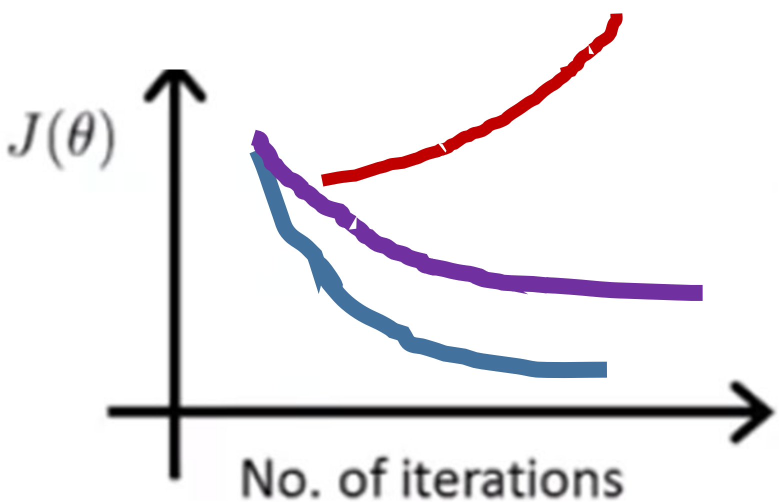
If x is a normal distribution, z is a standard normal distribution (0 mean and unit variance)

Making sure gradient descent is working properly

Proper use of learning curve







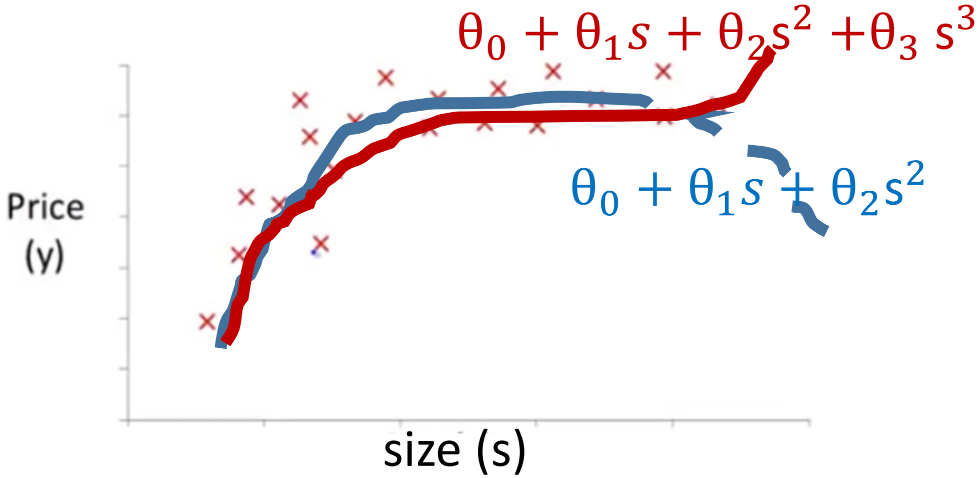
Select a range of learning rate

$\eta = 0.001,\ 0.003, 0.01,$
$0.03, \ldots$

Features and Polynomial Regression

(It is still linear regression but with polynomial features)

Polynomial Regression



Still consider a single physical feature: size (s).

Let :

x_1=(size) or x_1=s

x_2=(size)^2 or x_2= s^2

x_3=(size)^3 or x_3= s^3

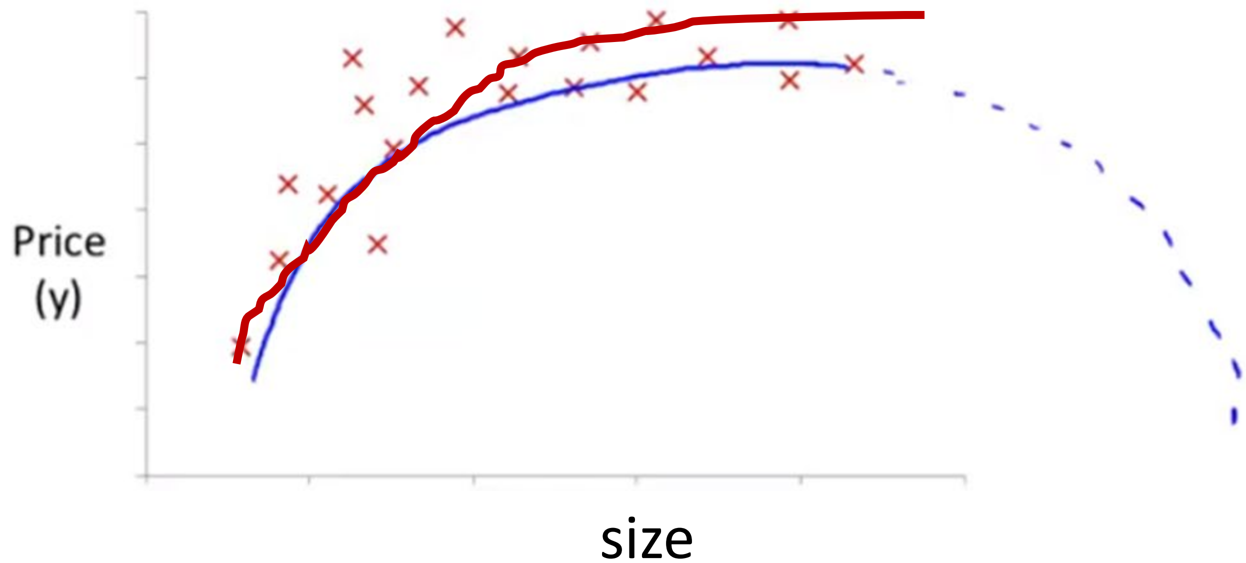
$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

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

Single variate polynomial regression as multi-variate linear regression

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$
$$= \theta_0 + \theta_1 s + \theta_2 s^2 + \cdots + \theta_n s^n$$

Choice of features



$$h_\theta(size) = \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2$$

$$h_\theta(size) = \theta_0 + \theta_1(\text{size}) + \theta_2\sqrt{(\text{size})}$$

Solve for θ

- Gradient descent
- Normal equation (closed form solution)

How to optimize (minimize) a cost/loss function by tuning neural network weights for a predicted value to approach an actual value?

- Stochastic gradient descent (such as the LMS by Widrow and Hoff)
- Batch gradient descent as in linear regression, quadratic problem with least squares solution in closed form (may not be the best for large data set)
- Mini-batches (as in many deep networks)

m **examples** $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ **;** n **features.**

Examples: $m = 4$.

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_4 x_4$$

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$$X\theta = y$$

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

m **training examples,** n **features.**

Gradient Descent	Normal Equation
• Need to choose η	• No need to choose η
• Needs many iterations.	• Don't need to iterate.

Use when n is large