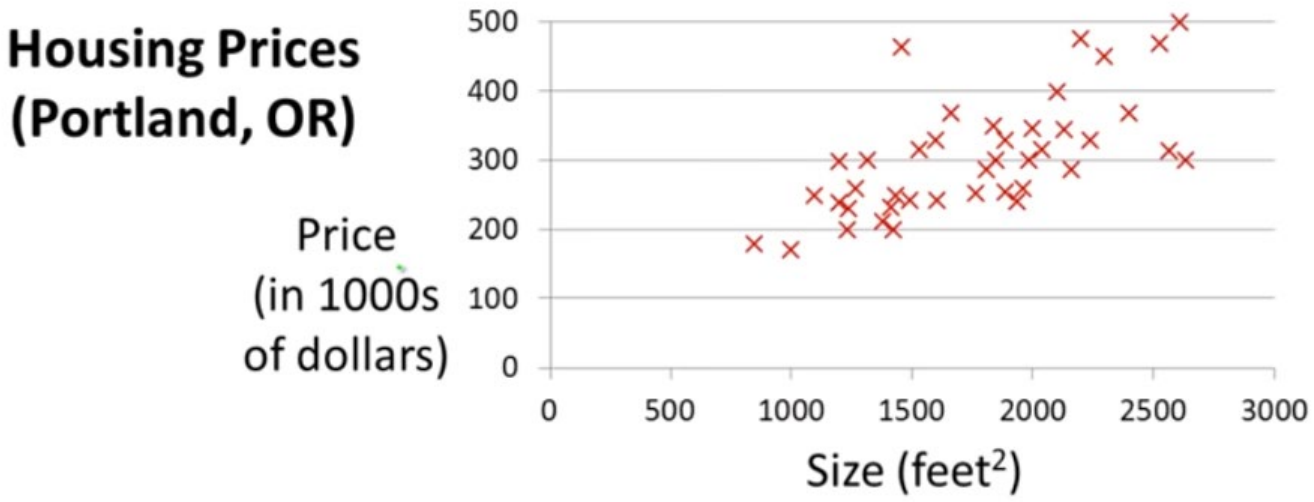
Univariate Linear Regression

Regression and machine learning

- Regression is a process of estimating the relationship between a dependent variable (or outcome variable) and independent variables (or features, predictors…)
- Linear regression – the dependent variable relates to the independent variables linearly.
- In the context of machine learning, regression is used for prediction/forecasting.



Supervised learning

Given data pairs (size, price), create parameterized models by supervised learning

Simplest such model, univariate linear regression:

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

Regression problem

Predict real valued output (price) given the size of a house just put on market

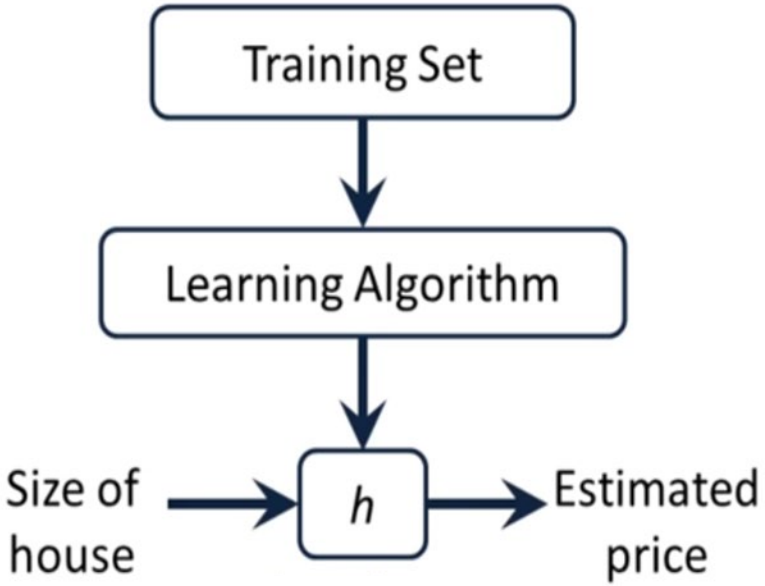
Classification problem

Predict discrete valued output (top moon/bottom moon) given the coordinates of a data point

Training set of housing prices (Portland, OR)

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

Notation:
m: number of training examples
x: input variable or input feature or feature
y: output variable



Hypothesis / predictive model

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

Univariate linear regression:

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

x: size of house

θ: parameter of model

How to choose model parameters θ_0 and θ_1?

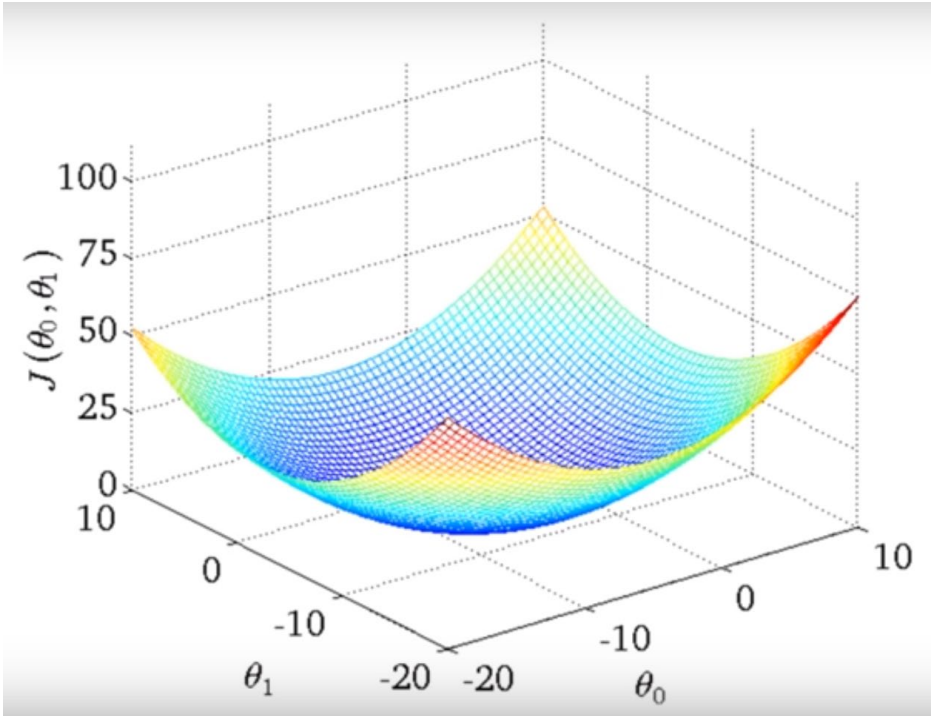
Goal: to make $h_\theta(x)$ as close to real house price (y) as possible (using the training data)

Problem formulation by first defining a cost function, $J(\theta_0, \theta_1)$:

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

Choose θ_0 and θ_1 so that is $J(\theta_0, \theta_1)$ minimized

Let's use examples to gain some intuition about the cost function $J(\theta_0, \theta_1)$
As a function of the parameters θ_0 and θ_1


100
75
50
25
0
$J(\theta_0, \theta_1)$
10
0
-10
-20
θ_1
-20
-10
0
10
θ_0

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

Parameters: θ_0 , θ_1

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

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Outline of learning procedure:

- Start with some θ_0 , θ_1
- Keep changing θ_0 , θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

Update parameters θ_0 and θ_1 using gradient descent until convergence

Simultaneously update θ_0 and θ_1 according to

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

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

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

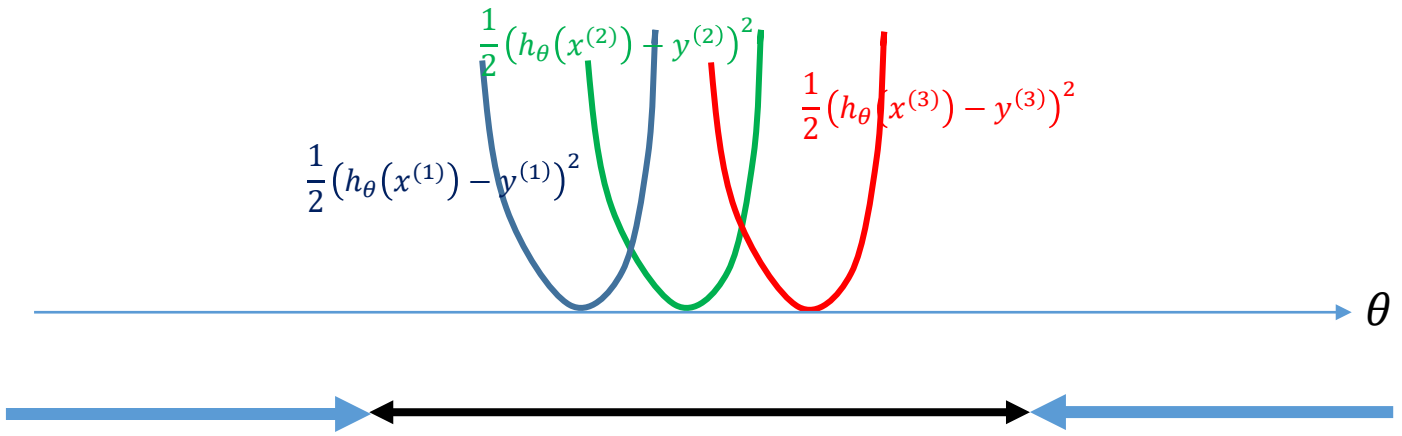
Exercise – derive the two gradients

- notice this derivation is a “batch” learning algorithm
- “online” learning – the LMS algorithm

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)

Insights on SGD (using Widrow-Hoff LMS setting)



Near convergence to solution of optimal filter, entering a region of confusion

- When far from convergence, behaves similarly as ordinary GD but less computation (1/m, m is the # of samples)
- When near convergence, gradient decent for one learning sample may be an ascent for another sample. A diminishing step size is required, e.g., the Robbins-Monro condition (e.g., η=1/n, n is # of learning iteration)