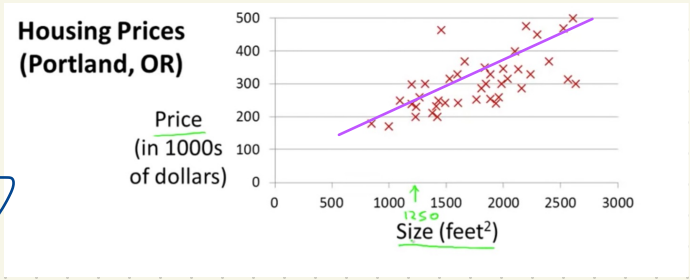
Model Representation.



supervised Learning (Given "right")

- Regression Problem.

Training set of housing prices (Portland, OR)

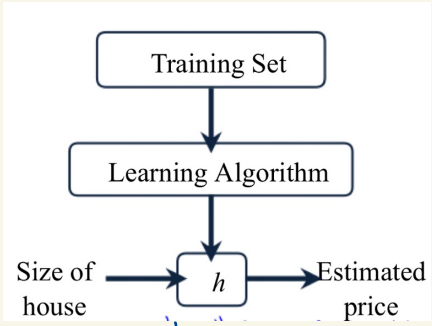
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

Notation:
- m = Number of training examples
- x's = "input" variable / features
- y's = "output" variable / "target" variable

eg: $x^{(1)} = 2104$
$x^{(2)} = 1416$
$y^{(1)} = 460$
$y^{(2)} = 232$

(x, y) : one training example.

$(x^{(i)}, y^{(i)})$: ith training example. (不是次方)



x hypothesis y (estimated value)

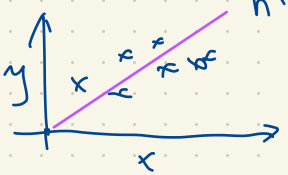
h is a function maps from x's to y's.

How do we represent h?

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

Short hands:

$h(x)$



$h(x) = \theta_0 + \theta_1 x$

we here we use linear func. (Continious).

Linear regression with one varible (x)

or call it

Univariate linear regression

↓

one variable.

Cost Function.

Find out the best fit.

Training Set

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

m=47

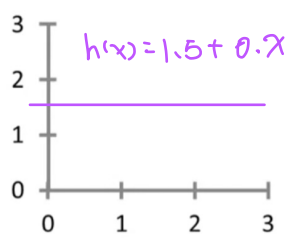
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_i's: Parameters.

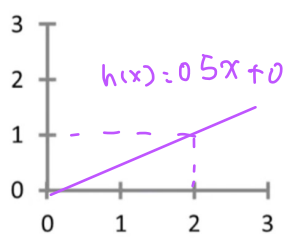
How to choose θ_i's?

With diff θ_i we get diff hypothesis (h(x))

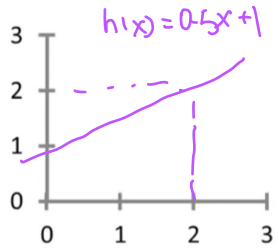
$h_\theta(x) = \theta_0 + \theta_1 x$



$\theta_0 = 1.5$
$\theta_1 = 0$



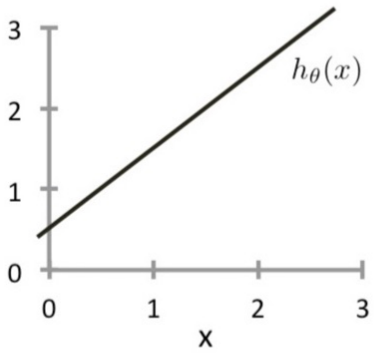
$\theta_0 = 0$
$\theta_1 = 0.5$



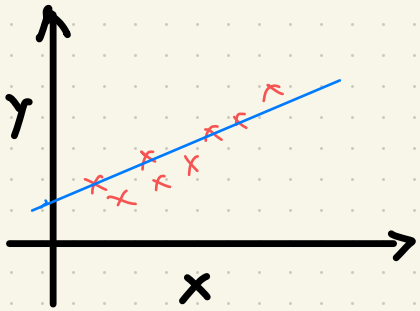
$\theta_0 = 1$
$\theta_1 = 0.5$

Question

Consider the plot below of $h_\theta(x) = \theta_0 + \theta_1 x$. What are θ_0 and θ_1?



- ○ $\theta_0 = 0, \theta_1 = 1$
- ✓ $\theta_0 = 0.5, \theta_1 = 1$
- ○ $\theta_0 = 1, \theta_1 = 0.5$
- ○ $\theta_0 = 1, \theta_1 = 1$



idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y).

It's eqv to the prob:

$$\underset{\theta_0\ \theta_1}{\text{minimize}} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

m → # of training examples (we num.)

$h_\theta(x^{(i)})$ → predicted y

$y^{(i)}$ → actual y

$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$

To mk the prob easier $\underset{\theta_0\ \theta_1}{\text{minimize}} \frac{1}{2m} \sum_{i=0}^{} (h_\theta(x^{(i)}) - y^{(i)})^2$

Get $\frac{1}{2m}$ size of the original, the min won't change.

Cost func:

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

$$\underset{\theta_0, \theta_1}{\text{minmize}}\ J(\theta_0, \theta_1)$$

squared error function

Cost Function – Intuition.

Hypothesis:

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

Parameters:

θ_0, θ_1

Cost Function:

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

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

→ 用 predict ($h(x)$) 减去 y！

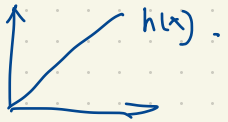
这是 actual

– Predict)

Predict – actual.

Simplified:

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



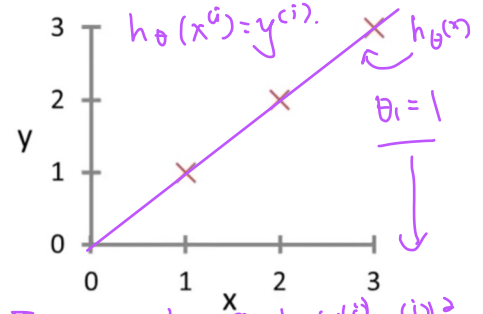
θ_1

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

$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

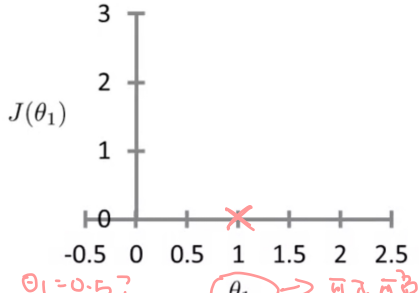
→ $h_\theta(x)$

(for fixed θ_1, this is a function of x)



→ $J(\theta_1)$

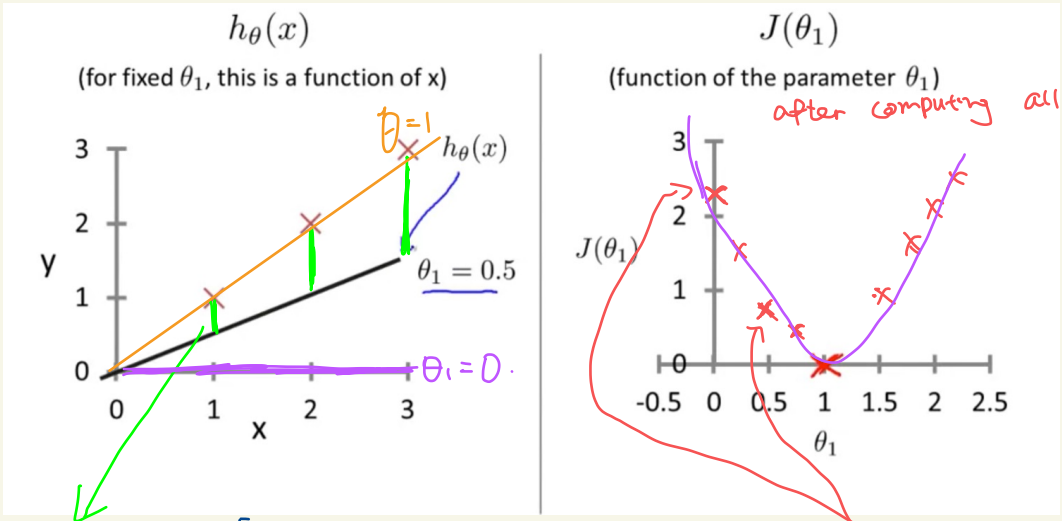
(function of the parameter θ_1)



$\theta_1 = 0.5$?

θ_1 → 可正可负

$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 = \frac{1}{2m}\left(\theta_1 x^{(1)} - y^{(1)}\right)^2 = \frac{1}{2m}(0^2 + 0^2 + 0^2) = 0^2 \Rightarrow J(1) = 0$

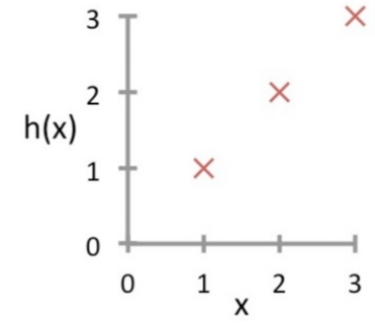


$J(0.5) = \frac{1}{2m}[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2]$

$= \frac{1}{2\times 3}(3.5) \approx 0.58.$

Question

Suppose we have a training set with m=3 examples, plotted below. Our hypothesis representation is $h_\theta(x) = \theta_1 x$, with parameter θ_1. The cost function $J(\theta_1)$ is $J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$. What is $J(0)$?



$h_\theta(x) = 0.$

$J(0) = \frac{1}{2\times 3}[(0-1)^2 + (0-2)^2 + (0-3)^2]$

$= \frac{1}{6}(1+4+9)$

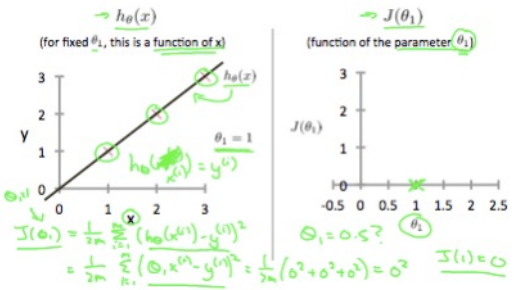
$= \frac{1}{6}(14) = \frac{14}{6}$

- 0
- 1/6
- 1
- ✓ 14/6

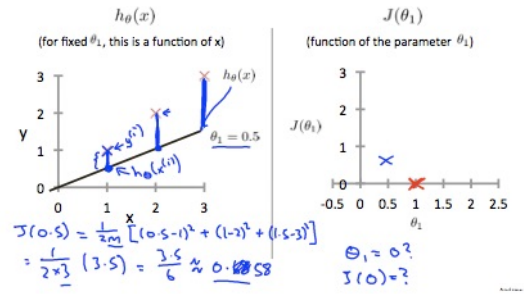
Cost Function - Intuition I

If we try to think of it in visual terms, our training data set is scattered on the x-y plane. We are trying to make a straight line (defined by $h_\theta(x)$) which passes through these scattered data points.

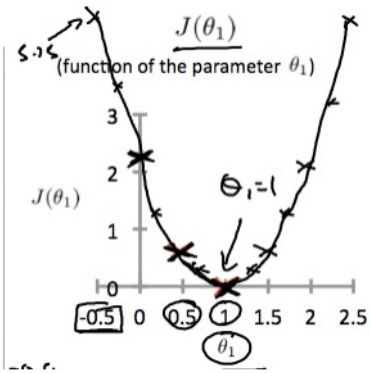
Our objective is to get the best possible line. The best possible line will be such so that the average squared vertical distances of the scattered points from the line will be the least. Ideally, the line should pass through all the points of our training data set. In such a case, the value of $J(\theta_0, \theta_1)$ will be 0. The following example shows the ideal situation where we have a cost function of 0.



When $\theta_1 = 1$, we get a slope of 1 which goes through every single data point in our model. Conversely, when $\theta_1 = 0.5$, we see the vertical distance from our fit to the data points increase.



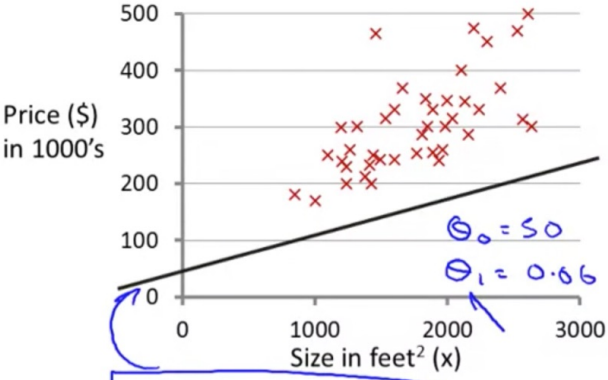
This increases our cost function to 0.58. Plotting several other points yields to the following graph:



Thus as a goal, we should try to minimize the cost function. In this case, $\theta_1 = 1$ is our global minimum.

$h_\theta(x)$

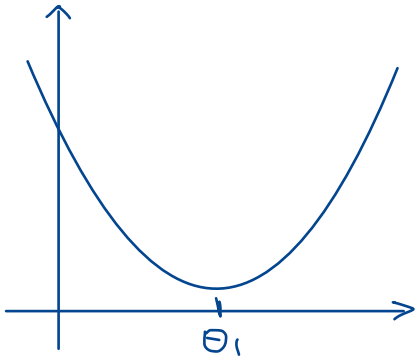
(for fixed θ_0, θ_1, this is a function of x)



$h_\theta(x) = 50 + 0.06x$

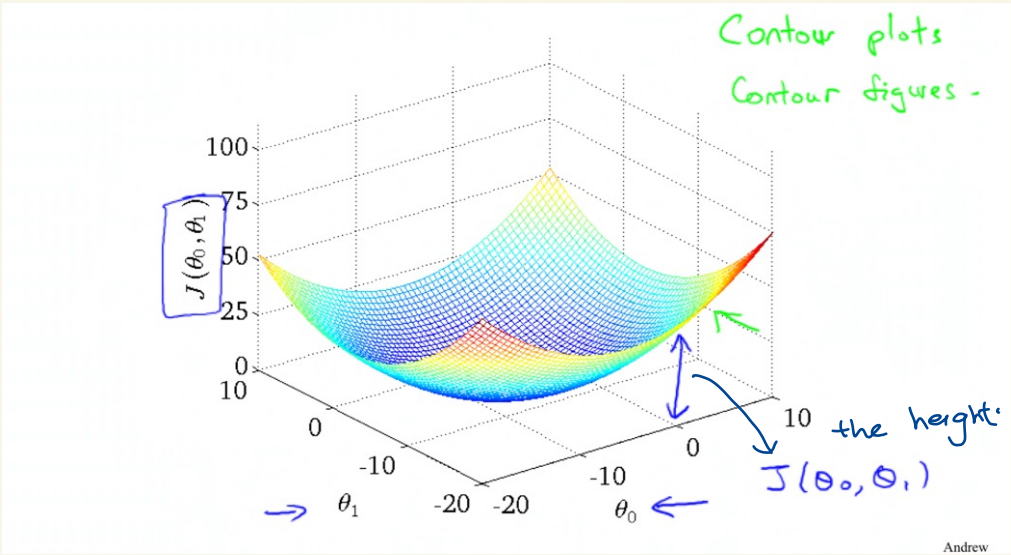
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



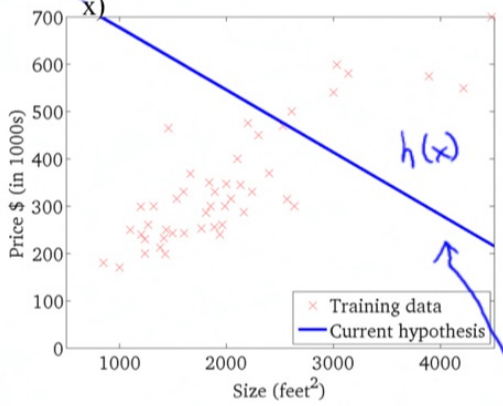
$\theta_0 = 0$.

$\theta_0 \neq 0$:



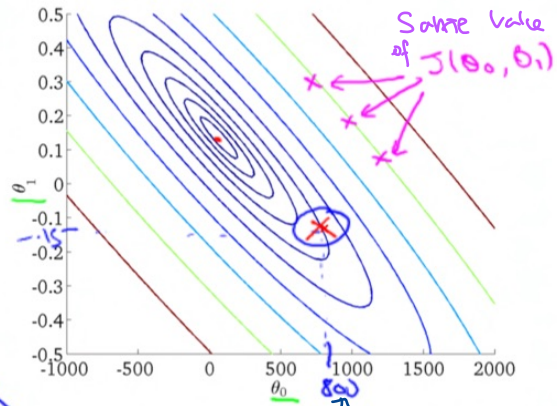
$h_\theta(x)$

(for fixed θ_0, θ_1 , this is a function of x)



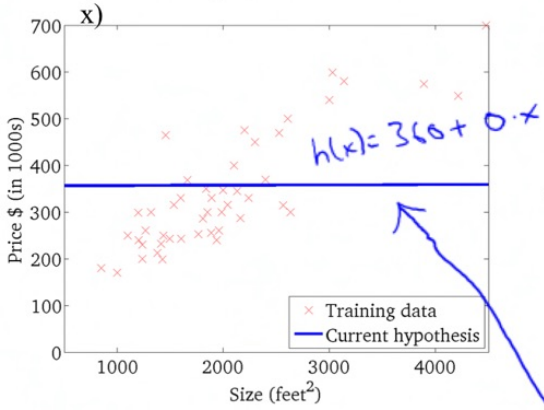
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



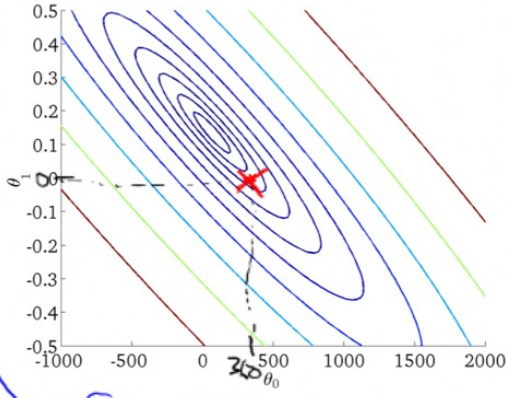
$h_\theta(x)$

(for fixed θ_0, θ_1 , this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

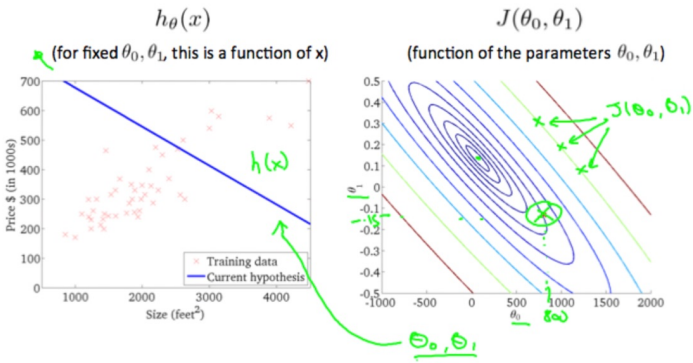


$\theta_0 = 360$

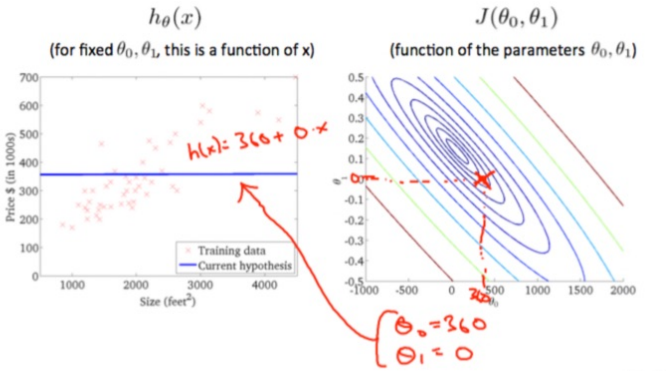
$\theta_1 = 0$

Cost Function - Intuition II

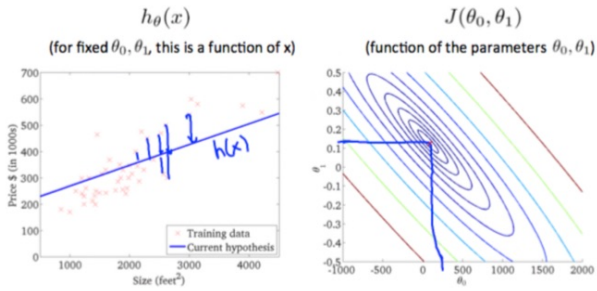
A contour plot is a graph that contains many contour lines. A contour line of a two variable function has a constant value at all points of the same line. An example of such a graph is the one to the right below.



Taking any color and going along the 'circle', one would expect to get the same value of the cost function. For example, the three green points found on the green line above have the same value for $J(\theta_0, \theta_1)$ and as a result, they are found along the same line. The circled x displays the value of the cost function for the graph on the left when θ_0 = 800 and θ_1 = -0.15. Taking another h(x) and plotting its contour plot, one gets the following graphs:



When θ_0 = 360 and θ_1 = 0, the value of $J(\theta_0, \theta_1)$ in the contour plot gets closer to the center thus reducing the cost function error. Now giving our hypothesis function a slightly positive slope results in a better fit of the data.



The graph above minimizes the cost function as much as possible and consequently, the result of θ_1 and θ_0 tend to be around 0.12 and 250 respectively. Plotting those values on our graph to the right seems to put our point in the center of the inner most 'circle'.

Gradient Descent.

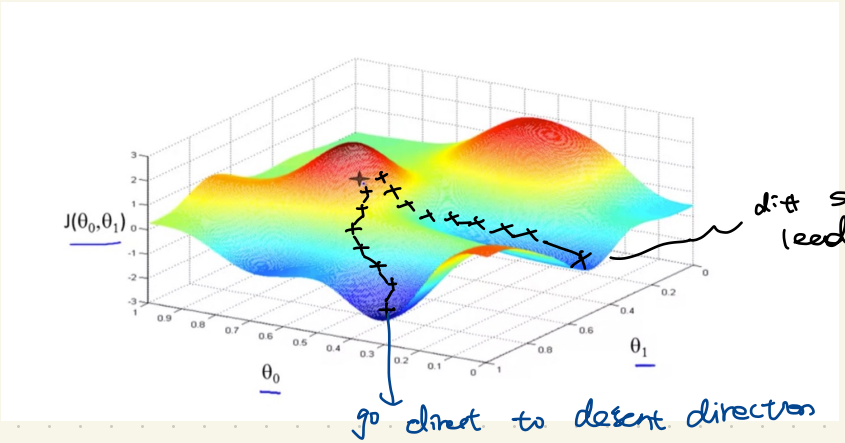
Have some function $J(\theta_0, \theta_1)$ $J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ $\min_{\theta_0 \cdots \theta_n} J(\theta_0, \ldots, \theta_n)$

Outline:

- Start with some θ_0, θ_1 (say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

until we hopefully end up at a minimum



Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$

}

learning rate (α↑ bigger step)

update θ_0 & θ_1 simultaneously

assignment
a := b (set a as b)

"=" is equal assertion
a = b (a is same as b).

a = a+1 ✗✗✗

Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Incorrect:

Not simultaneously

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$

θ_0 will be the new θ_0

Question

Suppose $\theta_0 = 1, \theta_1 = 2$, and we simultaneously update θ_0 and θ_1 using the rule: $\theta_j := \theta_j + \sqrt{\theta_0 \theta_1}$ (for j = 0 and j=1) What are the resulting values of θ_0 and θ_1?

- ○ $\theta_0 = 1, \theta_1 = 2$
- ✓ $\theta_0 = 1 + \sqrt{2}, \theta_1 = 2 + \sqrt{2}$
- ○ $\theta_0 = 2 + \sqrt{2}, \theta_1 = 1 + \sqrt{2}$
- ○ $\theta_0 = 1 + \sqrt{2}, \theta_1 = 2 + \sqrt{(1 + \sqrt{2}) \cdot 2}$

$\theta_0 = \theta_0 + \sqrt{2} = 1 + \sqrt{2}$

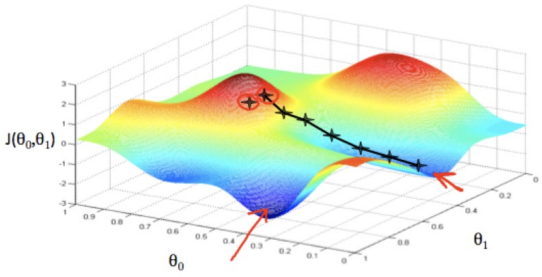
$\theta_1 = 2 + \sqrt{2}$

Gradient Descent

So we have our hypothesis function and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in the hypothesis function. That's where gradient descent comes in.

Imagine that we graph our hypothesis function based on its fields θ_0 and θ_1 (actually we are graphing the cost function as a function of the parameter estimates). We are not graphing x and y itself, but the parameter range of our hypothesis function and the cost resulting from selecting a particular set of parameters.

We put θ_0 on the x axis and θ_1 on the y axis, with the cost function on the vertical z axis. The points on our graph will be the result of the cost function using our hypothesis with those specific theta parameters. The graph below depicts such a setup.



We will know that we have succeeded when our cost function is at the very bottom of the pits in our graph, i.e. when its value is the minimum. The red arrows show the minimum points in the graph.

The way we do this is by taking the derivative (the tangential line to a function) of our cost function. The slope of the tangent is the derivative at that point and it will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter α, which is called the learning rate.

For example, the distance between each 'star' in the graph above represents a step determined by our parameter α. A smaller α would result in a smaller step and a larger α results in a larger step. The direction in which the step is taken is determined by the partial derivative of $J(\theta_0, \theta_1)$. Depending on where one starts on the graph, one could end up at different points. The image above shows us two different starting points that end up in two different places.

The gradient descent algorithm is:

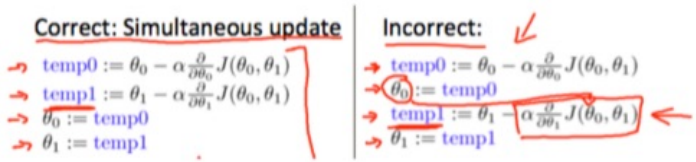
repeat until convergence:

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

where

j=0,1 represents the feature index number.

At each iteration j, one should simultaneously update the parameters $\theta_1, \theta_2, ..., \theta_n$. Updating a specific parameter prior to calculating another one on the $j^{(th)}$ iteration would yield to a wrong implementation.



Gradient Descent Intuition

Gradient descent algorithm

repeat until convergence {

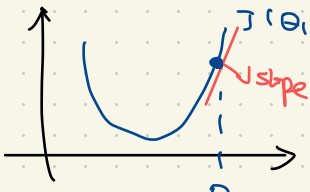
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(simultaneously update $j = 0$ and $j = 1$)

}

learning rate — derivative — determine the size of steps.

$\min_{\theta_1} J(\theta_1)$ $\quad \theta_1 \in \mathbb{R}$.

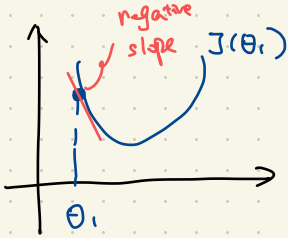


slope $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$

≥ 0.

$\theta_1 := \theta_1 - \alpha$ (positive num) $\Rightarrow$ move left

α always positive

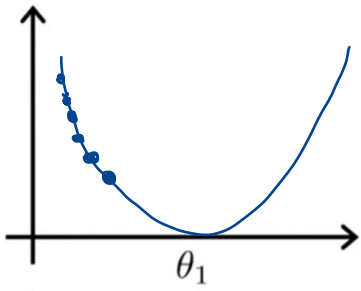


$\frac{d}{d\theta_1} J(\theta_1) \leq 0$

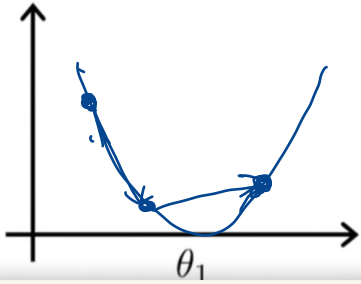
$\theta_1 := \theta_1 - \alpha$ (negative num) $\Rightarrow$ move right.

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.



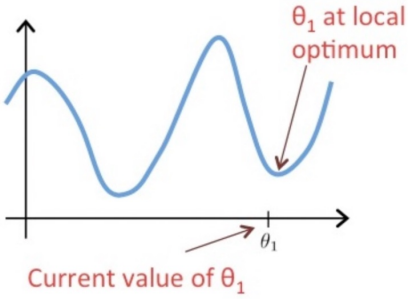
If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Question

Suppose θ_1 is at a local optimum of $J(\theta_1)$, such as shown in the figure.

What will one step of gradient descent $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$ do?



$\frac{d}{d\theta_1} J(\theta_1) = 0$

$\therefore \theta_1 := \theta_1 - \alpha \cdot 0$

$= \theta_1$

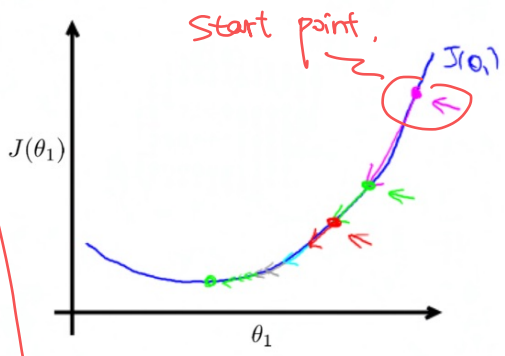
$\Rightarrow$ unchanged

- ✓ Leave θ_1 unchanged
- Change θ_1 in a random direction
- Move θ_1 in the direction of the global minimum of $J(\theta_1)$
- Decrease θ_1

Gradient descent can converge to a local minimum, even with the learning rate α fixed.

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.



due to the decrease of slope automatical decrease

=> auto take smaller steps

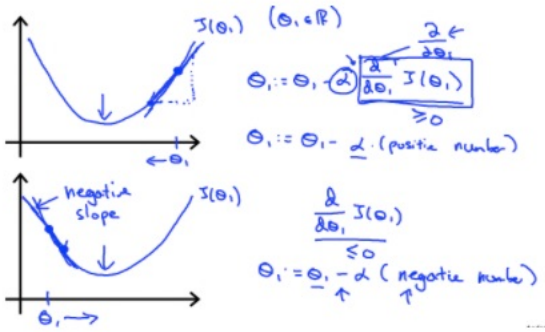
Gradient Descent Intuition

In this video we explored the scenario where we used one parameter θ_1 and plotted its cost function to implement a gradient descent. Our formula for a single parameter was :

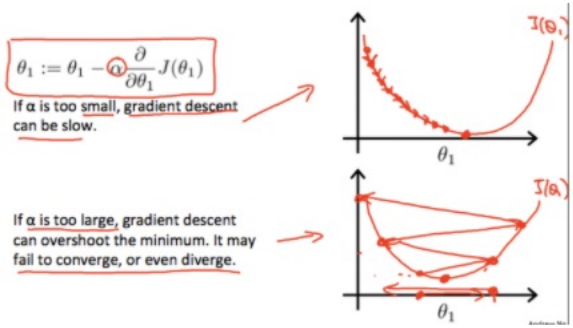
Repeat until convergence:

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Regardless of the slope's sign for $\frac{d}{d\theta_1} J(\theta_1)$, θ_1 eventually converges to its minimum value. The following graph shows that when the slope is negative, the value of θ_1 increases and when it is positive, the value of θ_1 decreases.



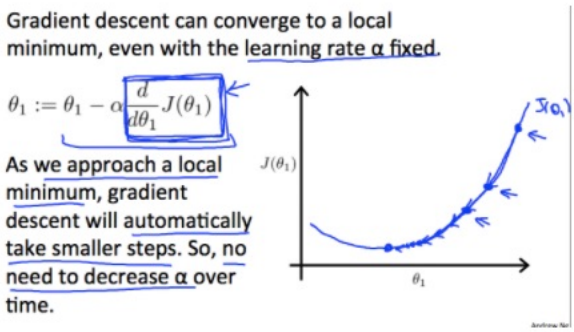
On a side note, we should adjust our parameter α to ensure that the gradient descent algorithm converges in a reasonable time. Failure to converge or too much time to obtain the minimum value imply that our step size is wrong.



How does gradient descent converge with a fixed step size α?

The intuition behind the convergence is that $\frac{d}{d\theta_1} J(\theta_1)$ approaches 0 as we approach the bottom of our convex function. At the minimum, the derivative will always be 0 and thus we get:

$$\theta_1 := \theta_1 - \alpha * 0$$



Gradient Desent For Leanre Regresia

Gradient descent algorithm

repeat until convergence {

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

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

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

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

$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

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

$$= \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$$

$$\theta_0 : j=0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

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

Gradient descent algorithm

repeat until convergence {

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

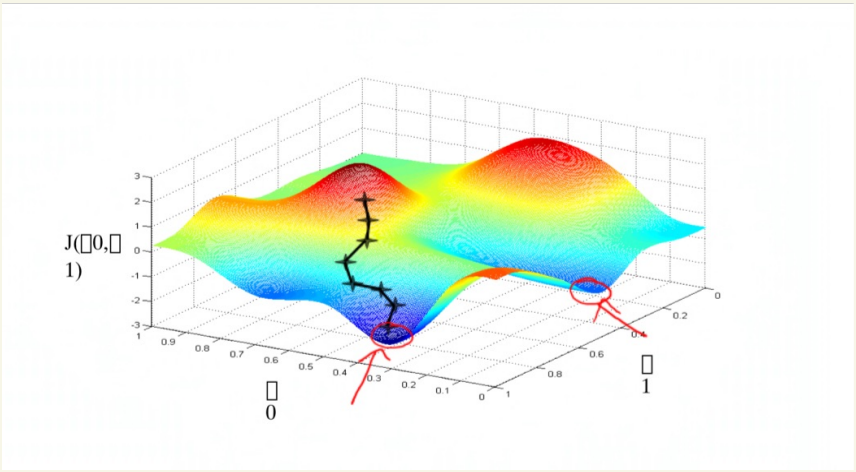
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$$

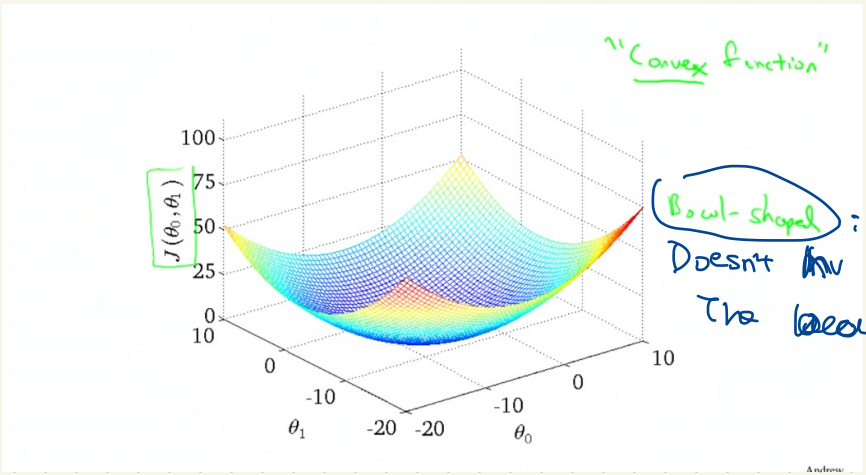
}

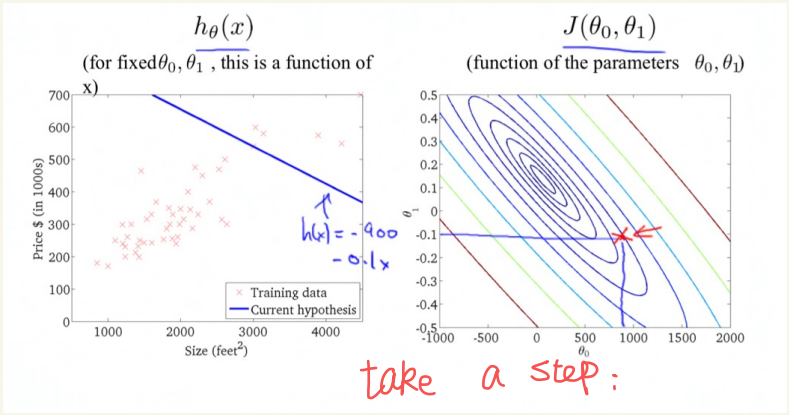
$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

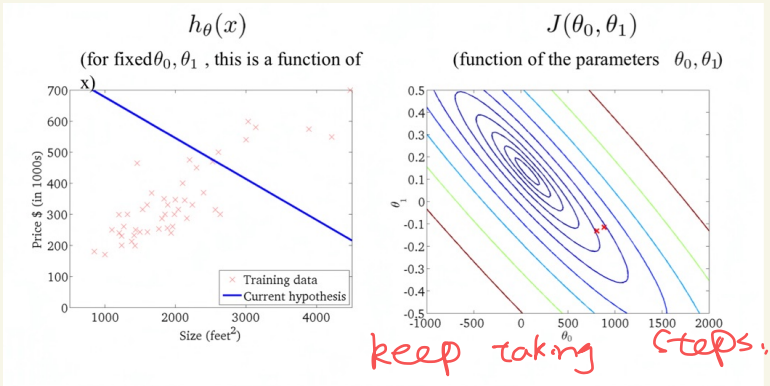
$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

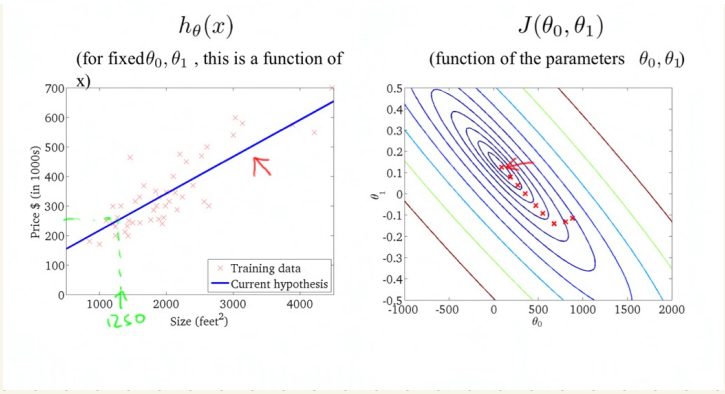
update θ_0 and θ_1 simultaneously


J(□0,□1)
□0
□1


"Convex function"
$J(\theta_0,\theta_1)$
100
75
50
25
0
10
0
-10
-20
θ_1
-20
-10
0
10
θ_0
Bowl-shaped:
Doesn't have local optimums
The local = global.


$h_\theta(x)$
(for fixed θ_0, θ_1 , this is a function of x)
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis
θ_0
θ_1
h(x) = -900 - 0.1x
take a step:


$h_\theta(x)$
(for fixed θ_0, θ_1 , this is a function of x)
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis
θ_0
θ_1
keep taking steps.


$h_\theta(x)$
(for fixed θ_0, θ_1 , this is a function of x)
$J(\theta_0, \theta_1)$
(function of the parameters θ_0, θ_1)
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis
1250
θ_0
θ_1

"Batch" Gradient Descent

"Batch": Each step of gradient descent uses all the training examples.

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

Question

Which of the following are true statements? Select all that apply.

- [ ] To make gradient descent converge, we must slowly decrease α over time.
- [ ] Gradient descent is guaranteed to find the global minimum for any function $J(\theta_0, \theta_1)$.
- [x] Gradient descent can converge even if α is kept fixed. (But α cannot be too large, or else it may fail to converge.)
- [x] For the specific choice of cost function $J(\theta_0, \theta_1)$ used in linear regression, there are no local optima (other than the global optimum).

Gradient Descent For Linear Regression

Note: [At 6:15 "h(x) = -900 - 0.1x" should be "h(x) = 900 - 0.1x"]

When specifically applied to the case of linear regression, a new form of the gradient descent equation can be derived. We can substitute our actual cost function and our actual hypothesis function and modify the equation to :

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) x_i)$$

}

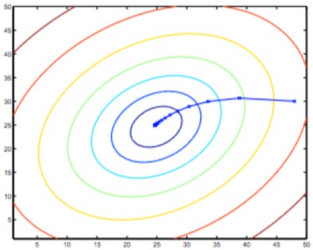
where m is the size of the training set, θ_0 a constant that will be changing simultaneously with θ_1 and x_i, y_i are values of the given training set (data).

Note that we have separated out the two cases for θ_j into separate equations for θ_0 and θ_1; and that for θ_1 we are multiplying x_i at the end due to the derivative. The following is a derivation of $\frac{\partial}{\partial \theta_j} J(\theta)$ for a single example :

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\
&= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\
&= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= (h_\theta(x) - y) \, x_j
\end{aligned}$$

The point of all this is that if we start with a guess for our hypothesis and then repeatedly apply these gradient descent equations, our hypothesis will become more and more accurate.

So, this is simply gradient descent on the original cost function J. This method looks at every example in the entire training set on every step, and is called **batch gradient descent**. Note that, while gradient descent can be susceptible to local minima in general, the optimization problem we have posed here for linear regression has only one global, and no other local, optima; thus gradient descent always converges (assuming the learning rate α is not too large) to the global minimum. Indeed, J is a convex quadratic function. Here is an example of gradient descent as it is run to minimize a quadratic function.



The ellipses shown above are the contours of a quadratic function. Also shown is the trajectory taken by gradient descent, which was initialized at (48,30). The x's in the figure (joined by straight lines) mark the successive values of θ that gradient descent went through as it converged to its minimum.