2. Model & Cost Function

2.1 Model Representation

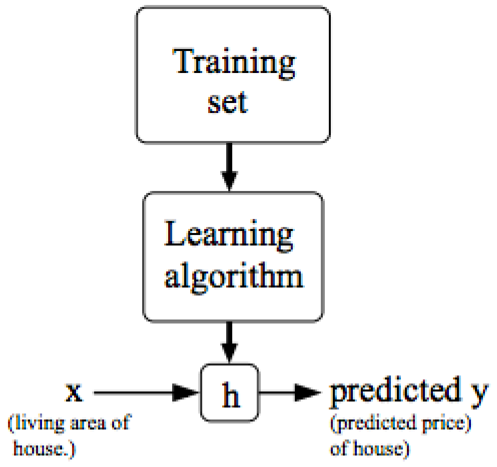
To establish notation for future use, we'll use $x^{(i)}$ to denote the **"input" variables** (living area in this example), also called input features, and $y^{(i)}$ to denote the **"output" or target variable** that we are trying to predict (price).

A pair $(x^{(i)}, y^{(i)})$ is called a **training example**, and the dataset that we'll be using to learn — i.e. a list of m training examples $(x^{(i)}, y^{(i)}); i = 1, \dots, m$ - is called a **training set.**

Note that the superscript "(i)" in the notation is simply an **index** into the training set, and has nothing to do with exponentiation.

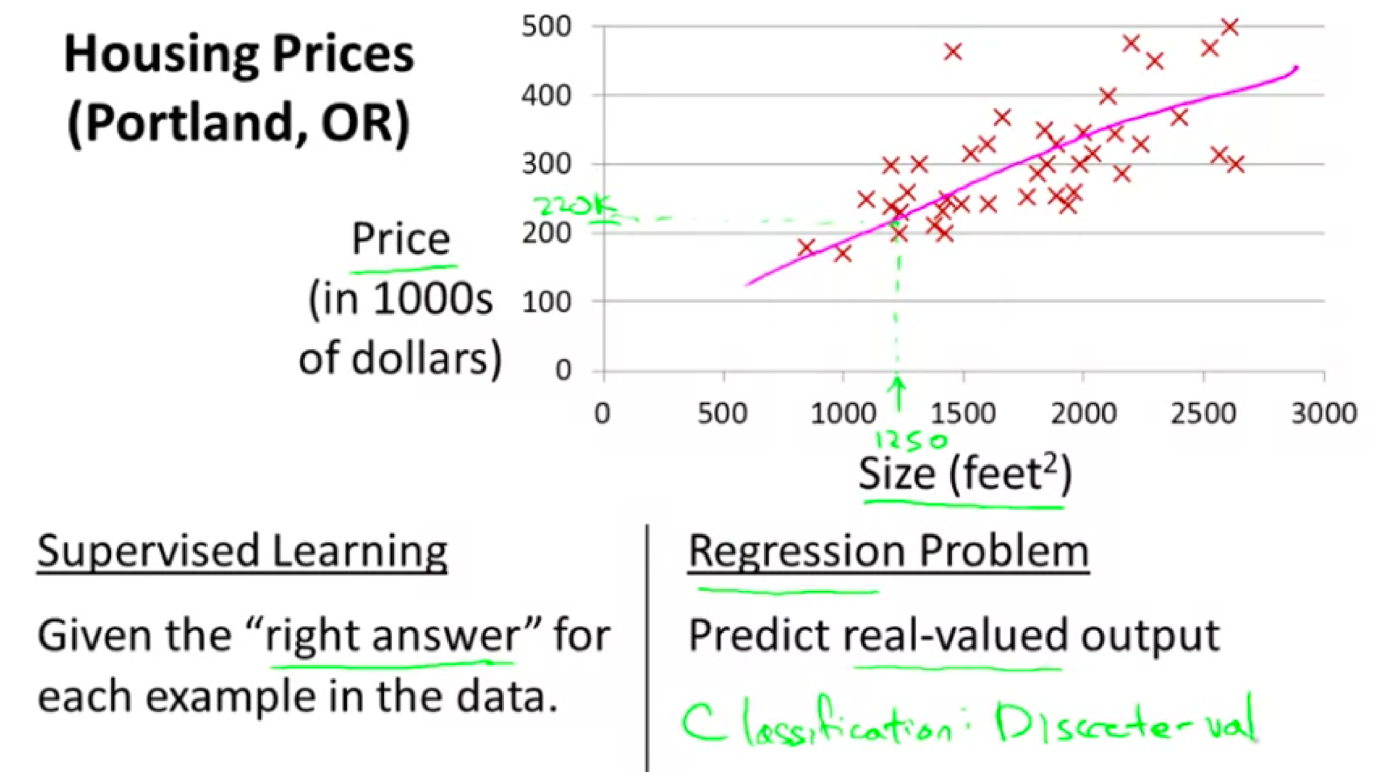
We will also use X to denote the space of input values, and Y to denote the space of output values. In this example, X = Y = $\mathbb{R}$.

To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function **h : X → Y** so that **h(x) is a "good" predictor for the corresponding value of y**. For historical reasons, this function h is called a **hypothesis.** Seen pictorially, the process is therefore like this:



When the target variable that we're trying to predict is continuous, such as in our housing example, we call the learning problem a regression problem. When y can take on only a small number of discrete values (such as if, given the living area, we wanted to predict if a dwelling is a house or an apartment, say), we call it a classification problem.

Example: prices of houses in Portland, given their size in squared feet



Supervised Learning

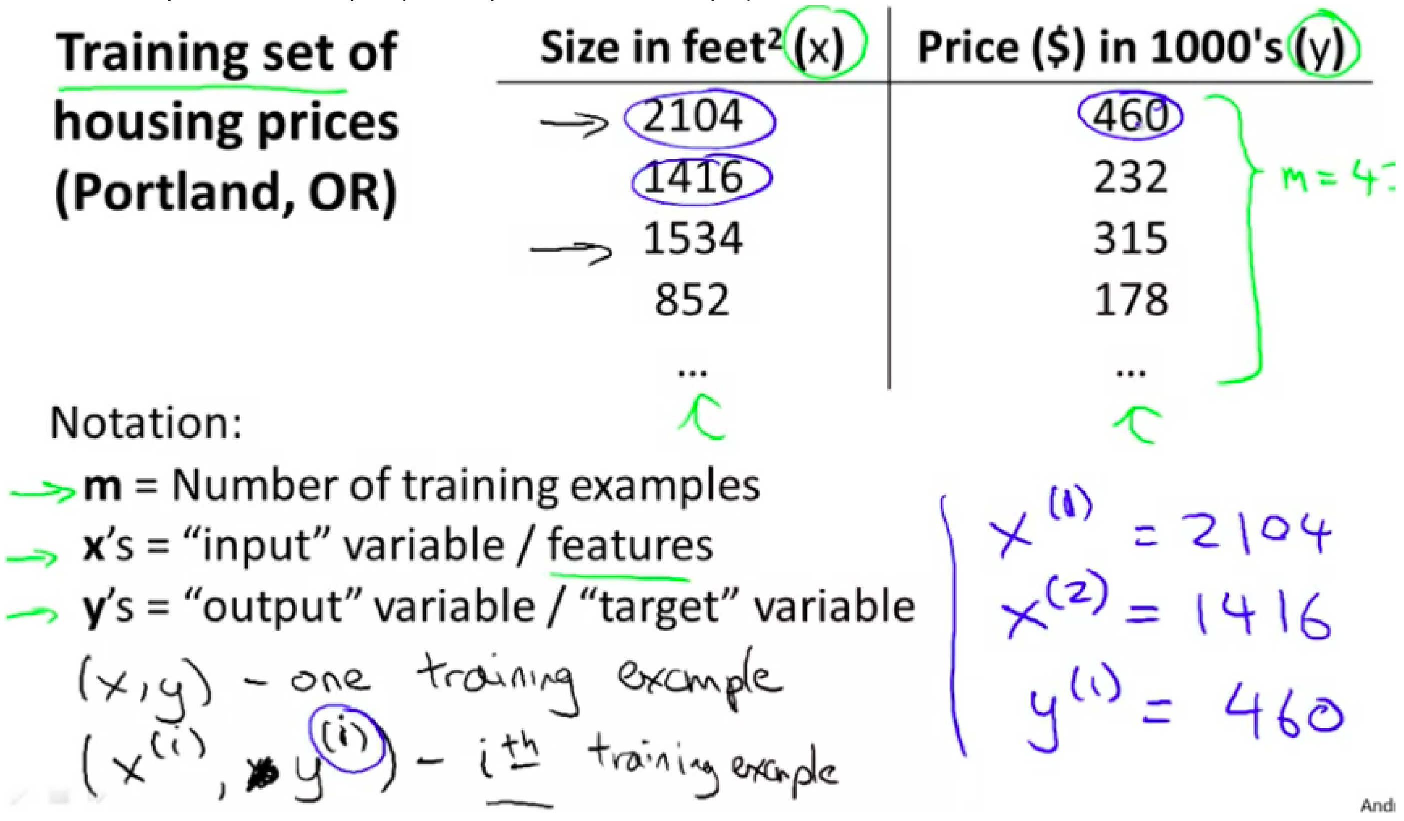
Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Classification: Discrete-val

More formally, in supervised learning we have a dataset, called training set, from which the algorithm needs to learn how to predict the output (house price in this example).



Training set of housing prices (Portland, OR)

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

m=4

Notation:

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

(x, y) - one training example

$(x^{(i)}, y^{(i)})$ - i^{th} training example

$x^{(1)} = 2104$

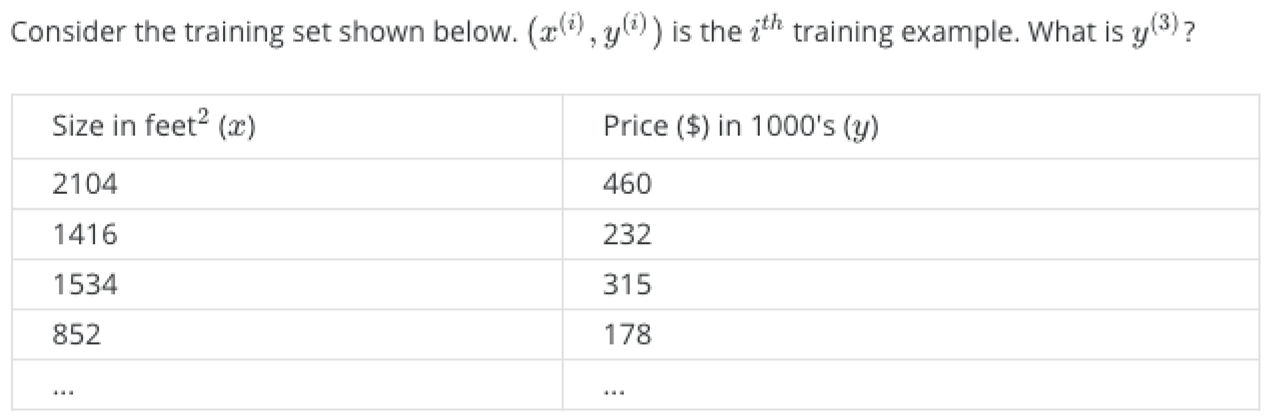
$x^{(2)} = 1416$

$y^{(1)} = 460$

Andi

- **m** is the number of rows in the training set.
- The superscript (i) refers to the i-th row in the training set table.

Question:

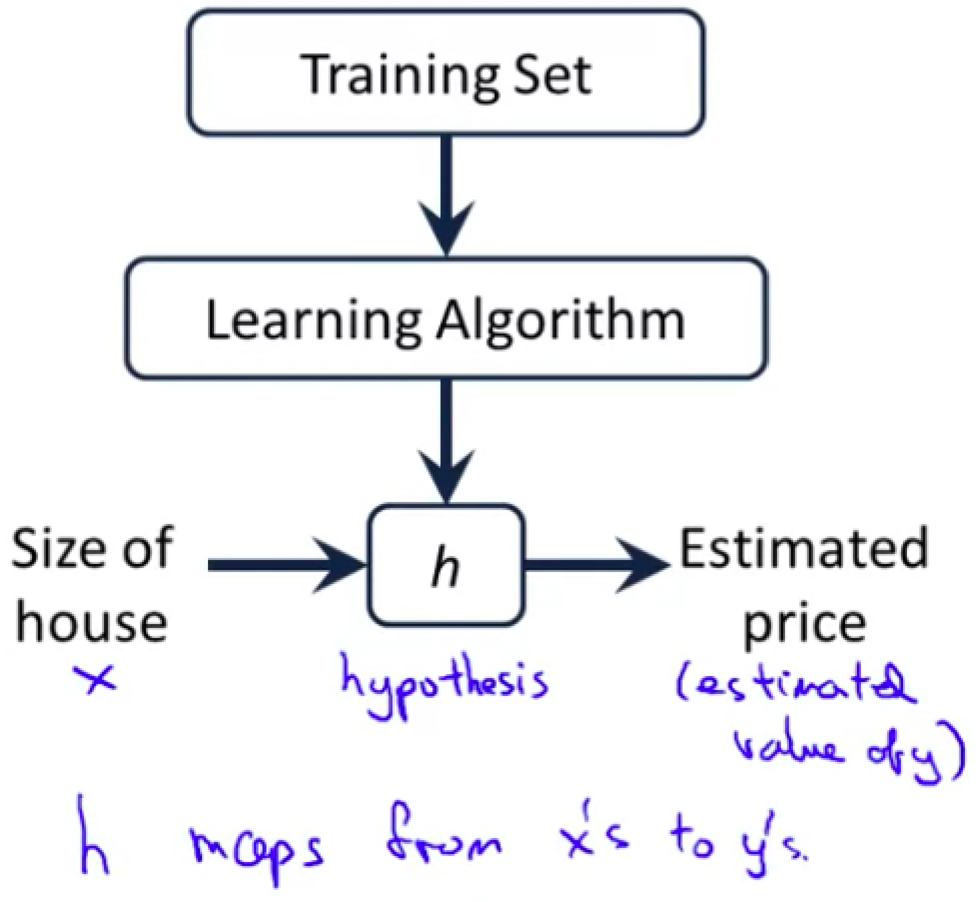


Consider the training set shown below. $(x^{(i)}, y^{(i)})$ is the i^{th} training example. What is $y^{(3)}$?

Size in feet^2 (x)	Price ($) in 1000's ($y$)
2104	460
1416	232
1534	315
852	178
...	...

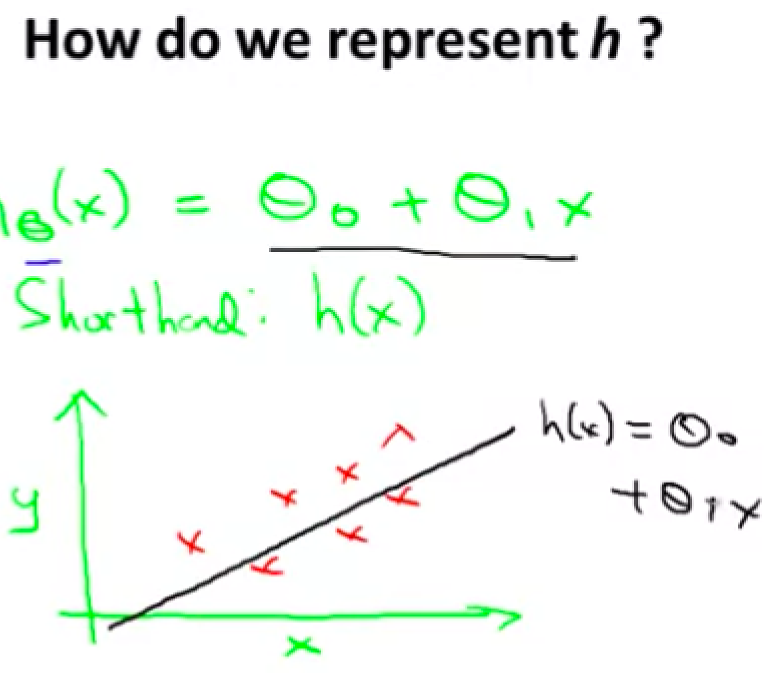
$y^{(3)}$ = 315

The training set is fed to an algorithm whose job is to provide as output a function (h) that takes as input the dataset features (e.g. house size) and gives as output the target variable (e.g. house estimated price):



When designing a learning algorithm, the next thing we need to decide is how do we represent the hypothesis function h.

For this example, we estimate y is a linear function of x:


How do we represent *h* ?
$h_\theta(x) = \theta_0 + \theta_1 x$
Shorthand: $h(x)$
$h(x) = \theta_0 + \theta_1 x$
y
x

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

This model is called **linear regression with one variable** or **univariate linear regression**.

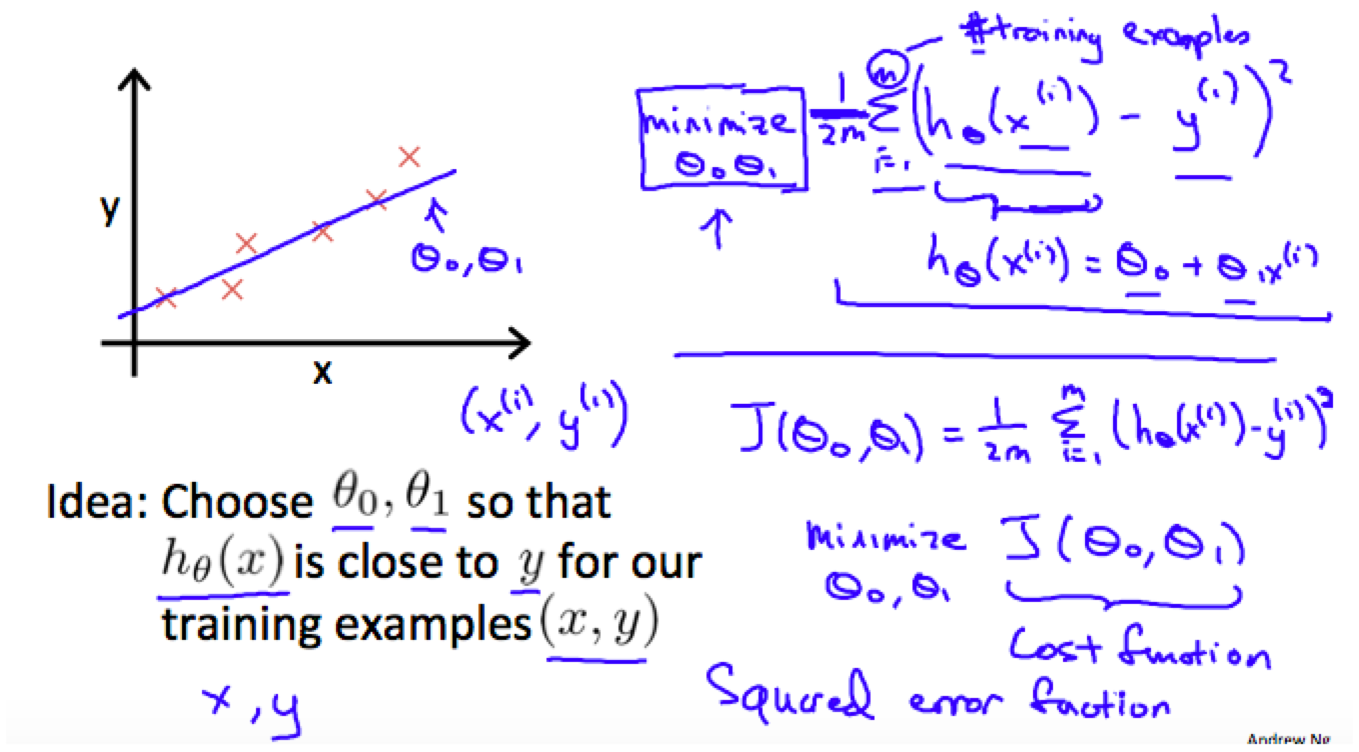
2.2 Cost Function

We can measure the accuracy of our hypothesis function by using a cost function. This takes an average difference (actually a fancier version of an average) of all the results of the hypothesis with inputs from x's and the actual output y's:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

To break it apart, it is $\frac{1}{2}\bar{x}$, where $\bar{x}$ is the mean of the squares of $h_\theta(x_i) - y_i$, or the difference between the predicted value and the actual value.

This function is otherwise called the **"Squared error function"**, or **"Mean squared error".** The mean is halved as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the $\frac{1}{2}$ term. The following image summarizes what the cost function does:



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

m — # training examples

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

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

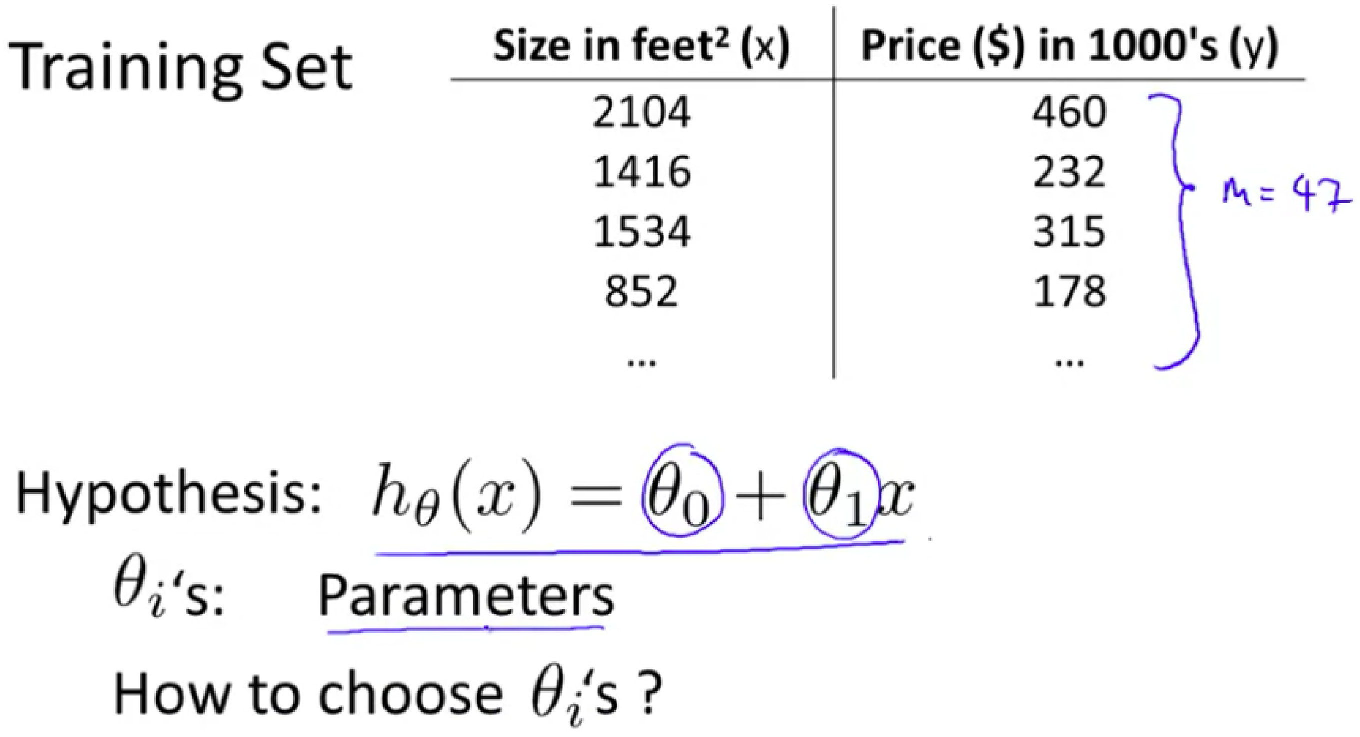
Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

x, y

$$\underset{\theta_0,\theta_1}{\text{minimize}} \ \underbrace{J(\theta_0,\theta_1)}_{\text{Cost function}}$$

Squared error function

In our example, the cost function let's us figure out how to fit the best possible regression line to our data:



Training Set

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

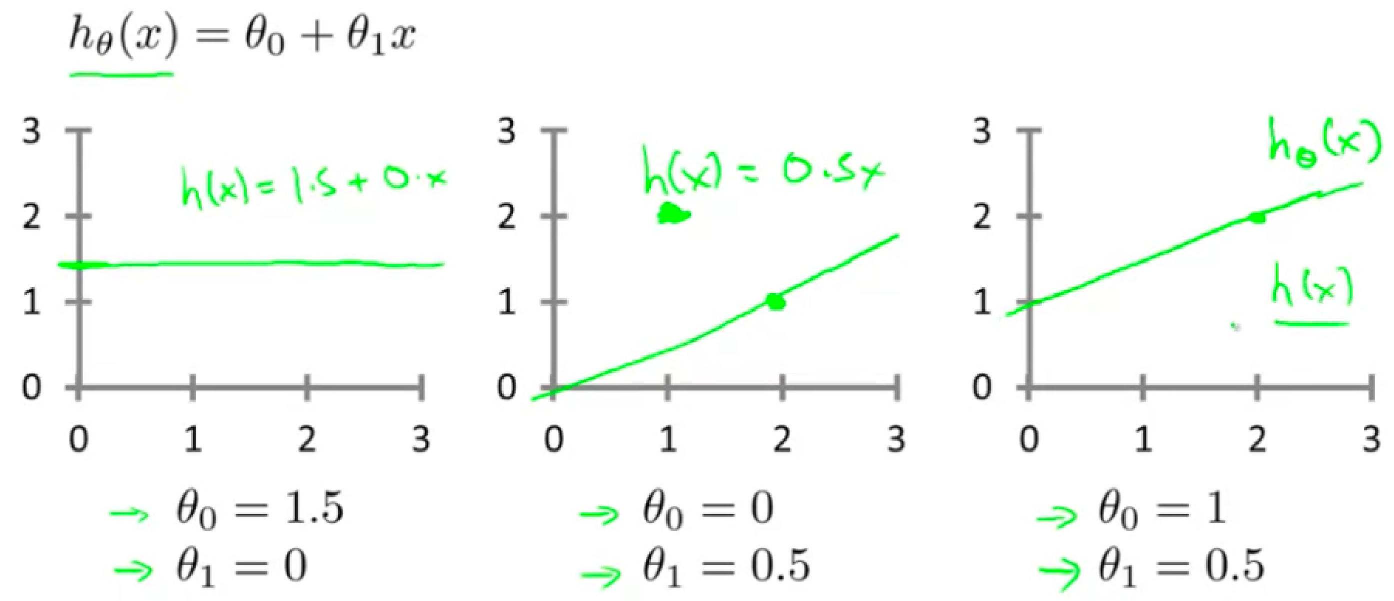
$m = 47$

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

θ_i's: Parameters

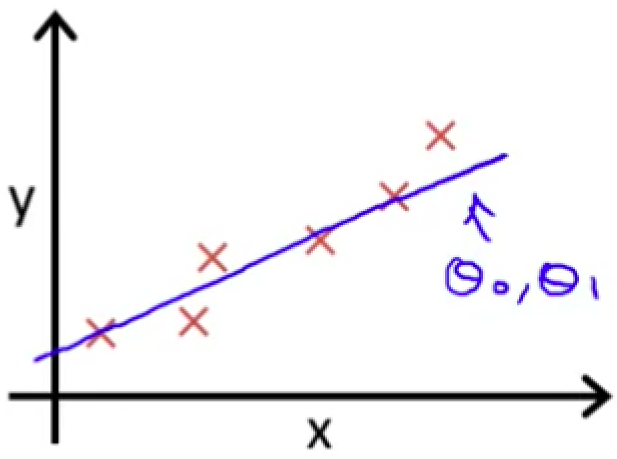
How to choose θ_i's ?

With different choices of the parameters θ_0 and θ_1, we generate different hypothesis functions:

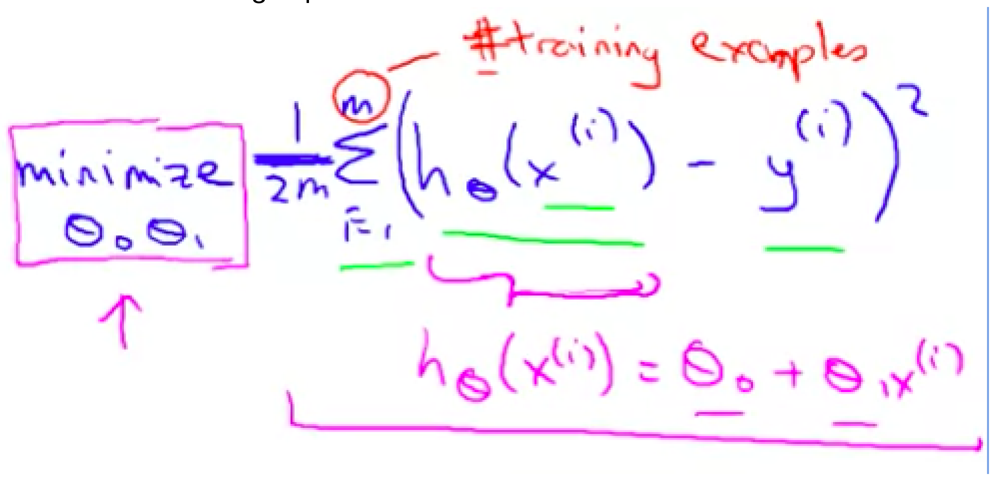


In linear regression, we want to use our training set to find values for the parameters θ_0 and θ_1 so that the resulting straight line from the h function fits the data well. How do we find the values of the parameters?

The idea is to choose θ_0, θ_1 so that $h(x)$ is close to y for our training examples (x, y)



In other words, in linear regression we want to solve a minimisation problem, i.e. find the value of the parameters that minimise the following expression:



#training examples
$\underset{\theta_0, \theta_1}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$

This is equivalent to minimise the **cost function** $J(\theta_0, \theta_1)$:

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

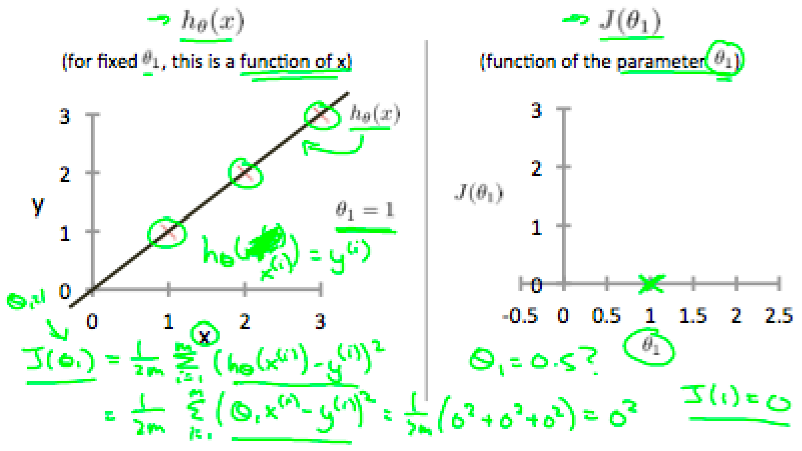
$$\underset{\theta_0,\theta_1}{\text{Minimize}}\ \underbrace{J(\theta_0,\theta_1)}_{\text{Cost function}}$$

The cost function is also called the **squared error function**. This function is a resonable choice and works well for most regression problems. However, there are also other cost functions that would work pretty well, but the squared error function is commonly used.

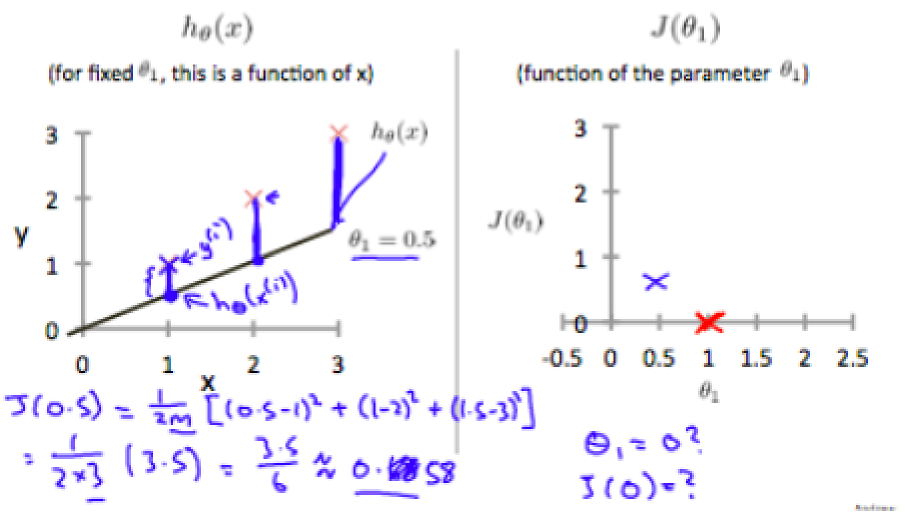
2.3 Cost Function - Intuition I

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 (simplification with $J(\theta_1)$):



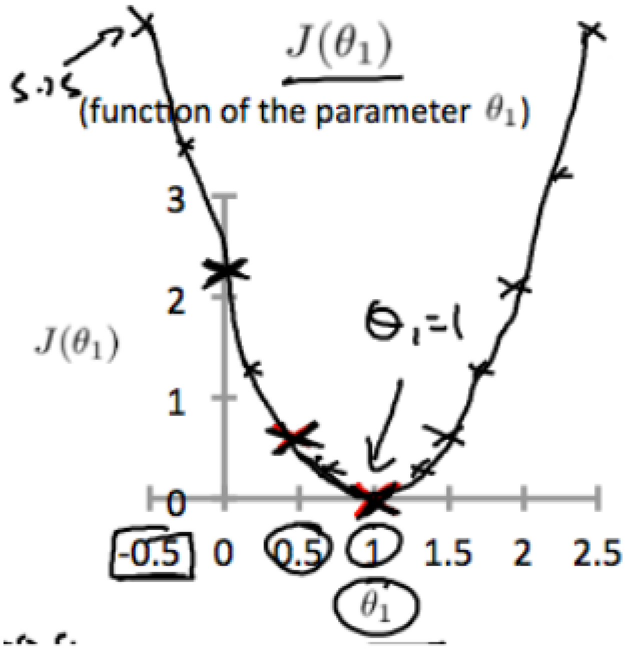
When $\theta_1 = 1$, we get a slope of 1 which goes through every single data point in our model. Conversely, when $\theta_1 = 0.5$, we see the vertical distance from our fit to the data points increase. This increases our cost function to 0.58.



When $\theta_1 = 0$,

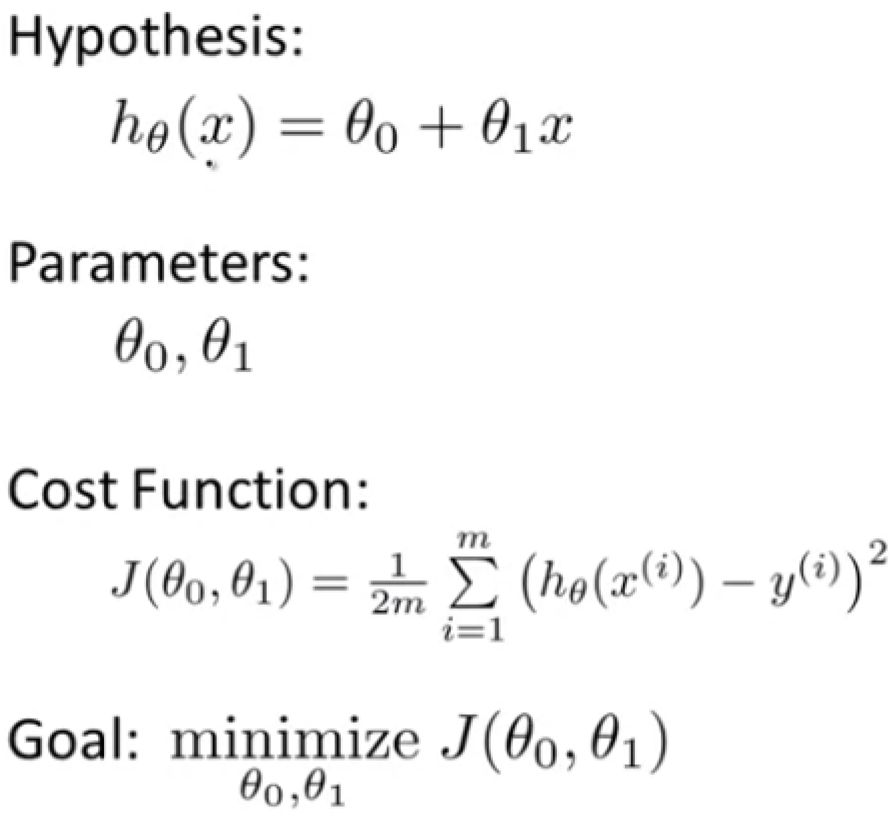
$$J(0) = \frac{1}{6}[(0-1)^2 + (0-2)^2 + (0-3)^2] = \frac{1}{6}14 = \frac{7}{3} \approx 2.333$$

Plotting several other points yields to the following graph:



Thus as a goal, we should try to minimize the cost function. In this case, θ_1 is our global minimum.

To recap (the final goal is the minimisation of the cost function $J(parameters)$):



Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

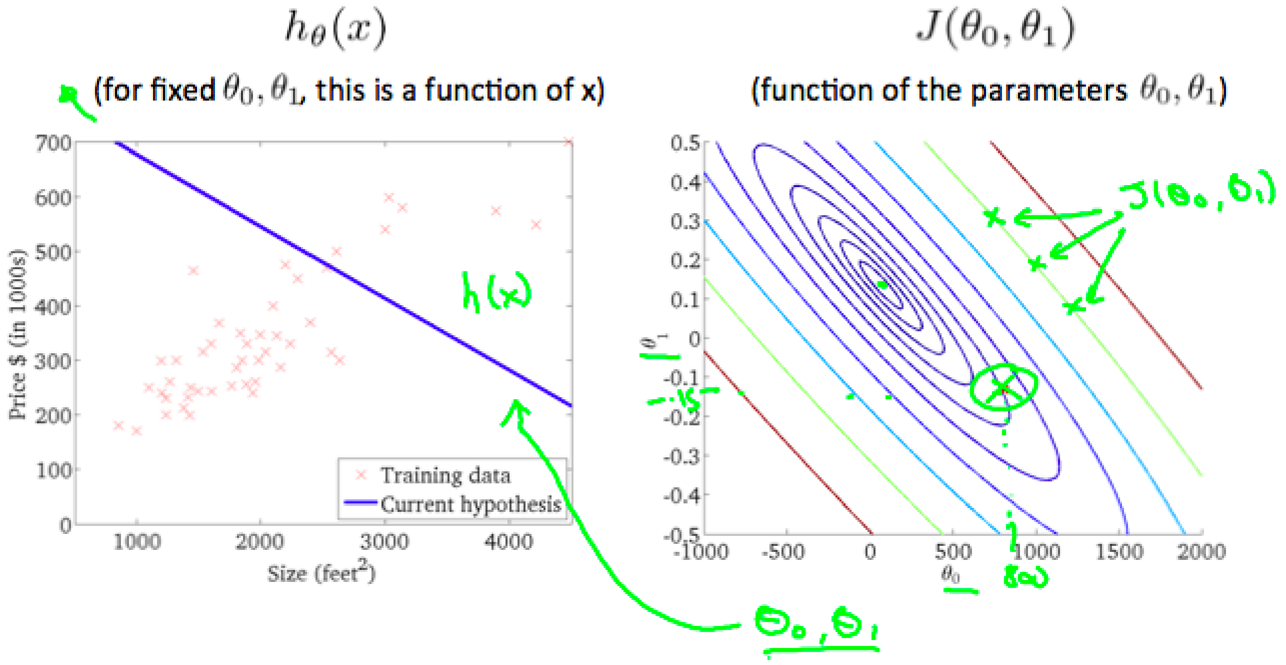
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

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

Each combination of values of the parameters corresponds to a different value of the cost function.

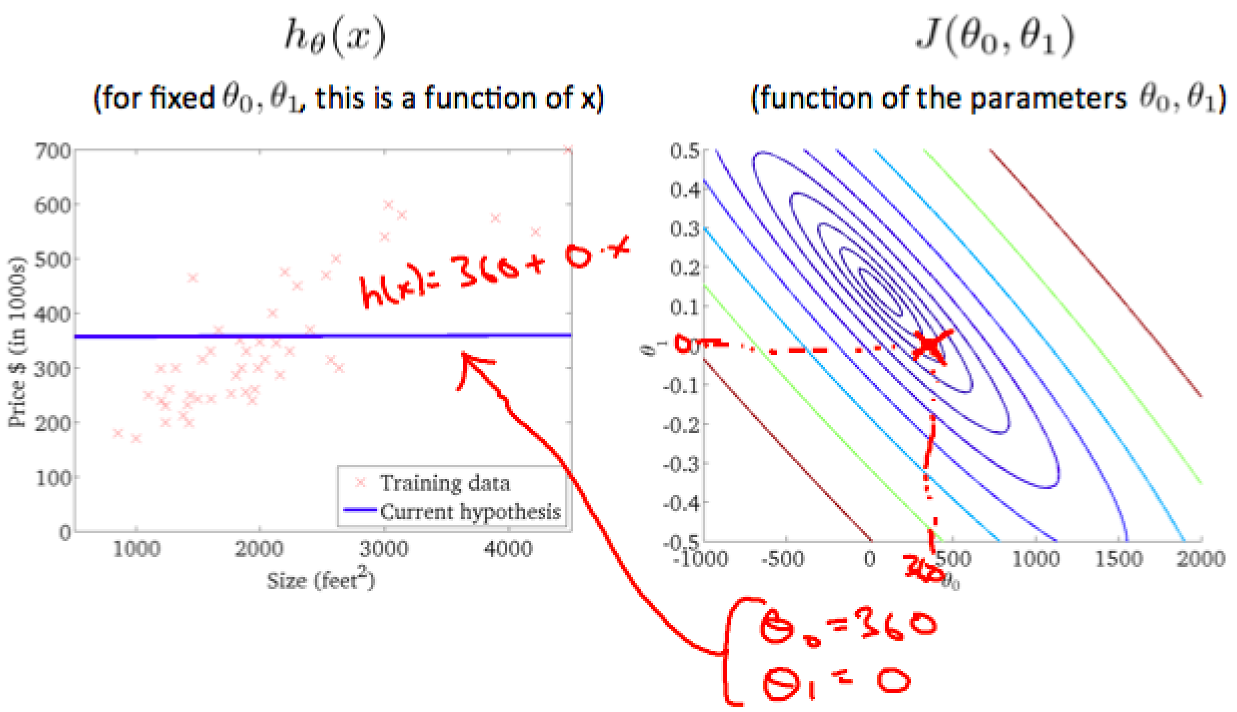
2.4 Cost Function - Intuition II

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.



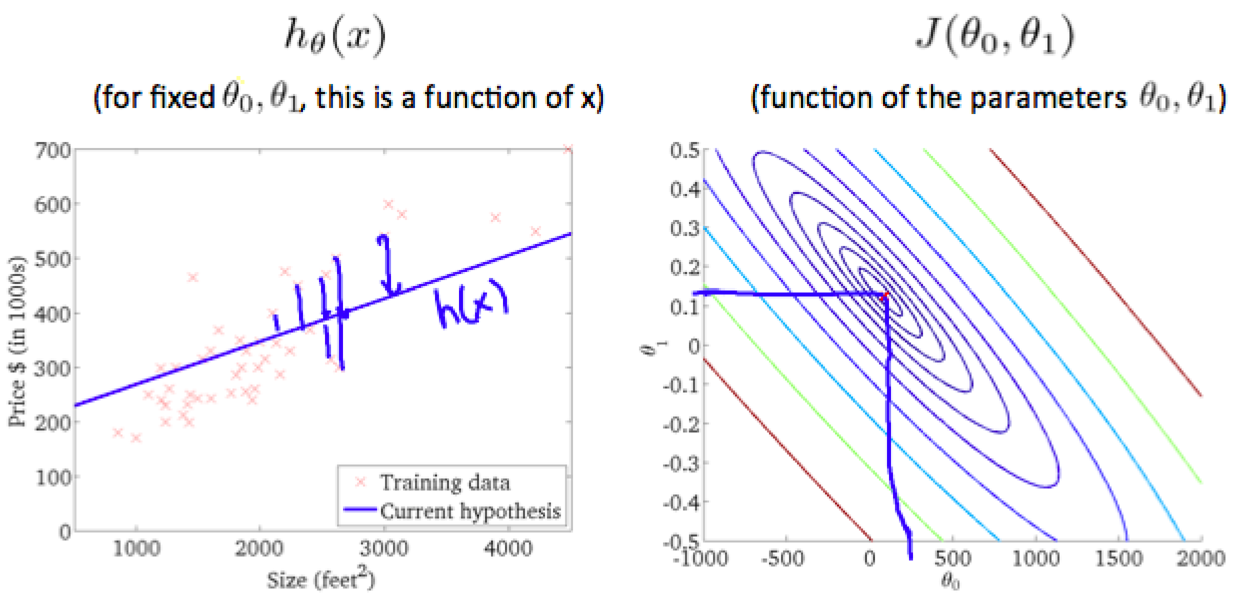
Taking any color and going along the 'circle', one would expect to get the same value of the cost function. For example, the three green points found on the green line above have the same value for $J(\theta_0, \theta_1)$ and as a result, they are found along the same line. The circled x displays the value of the cost function for the graph on the left when $\theta_0 = 800$ and $\theta_1 = -0.15$. (This is equivalent to say: $h(x) = -0.15x + 800$)

Taking another $h(x)$ and plotting its contour plot, one gets the following graphs:



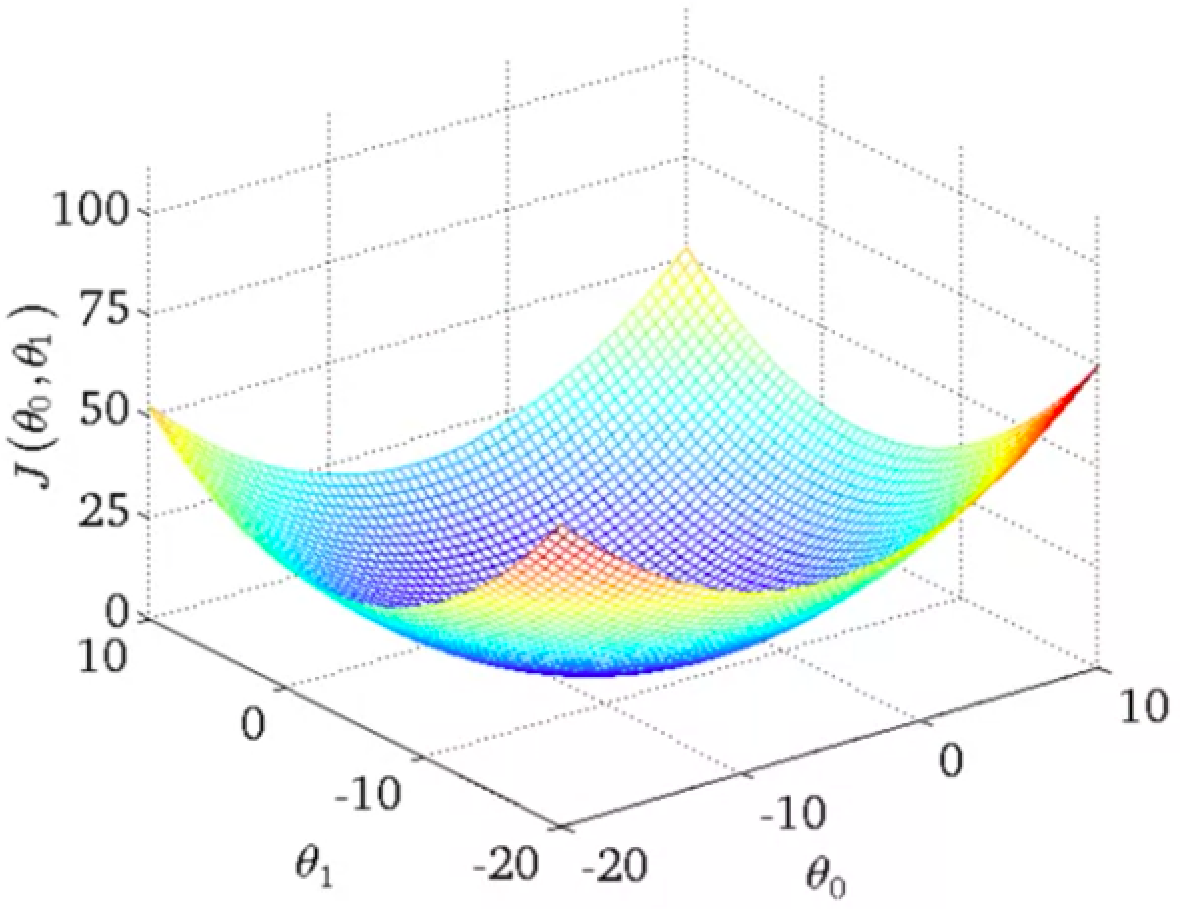
When $\theta_0 = 360$ and $\theta_1 = 0$, the value of $J(\theta_0, \theta_1)$ in the contour plot gets closer to the center thus reducing the cost function error.

Now giving our hypothesis function a slightly positive slope results in a better fit of the data ($h(x) = 0.12x + 250$).



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'.

Another visualisation of the cost function in three dimension, having 2 parameters (in this simple case, given the training dataset, the 3-dimensional surface is bowl-shaped):



What we are looking for is an efficient algorithm to automatically find the values of the parameters that fit best the model to our data (i.e. those that minimise the cost function).

When you look at more complicated data you end with high-dimensional figures that cannot be visualised.