Last lectures: Linear regression with one variable

- Overall process of supervised learning
- How the model is represented
- Cost function: provide the best fit model
  - **deeper look (Intution- I)**
  - **Intuition-2**
- Gradient descent representations:
  - Graphical
  - Mathematically
- Understanding the Gradient descent for Linear regression

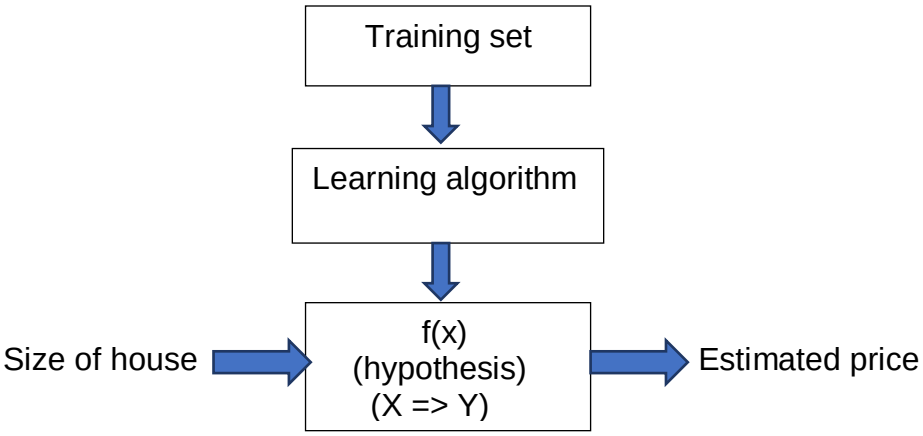
1) What is overall process of machine learning?

- Training set of housing price of a city (this is your data set)

Size in feet square (x)	Prince ($) in 1000's (y)
2014	460
1416	232
1534	315
852	178
….	….

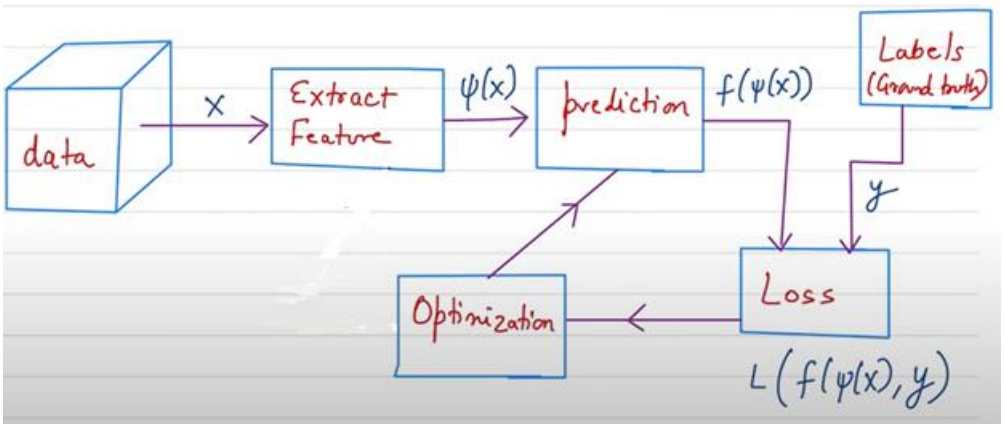
- **Notation**
  - m = number of **training examples** = 47
  - x's = input variables / features
    - (x,y) - single training example
    - $(x^{(i)}, y^{(i)})$ - specific example (i^{th} training example)
      - i is an index to training set
      - if i = 1=> (2014,460)
  - y's = output variable "target" variables
  - n = 1 = no of input variables/features/attributes (size of house/ no of room)

Now how the supervising algorithm works:



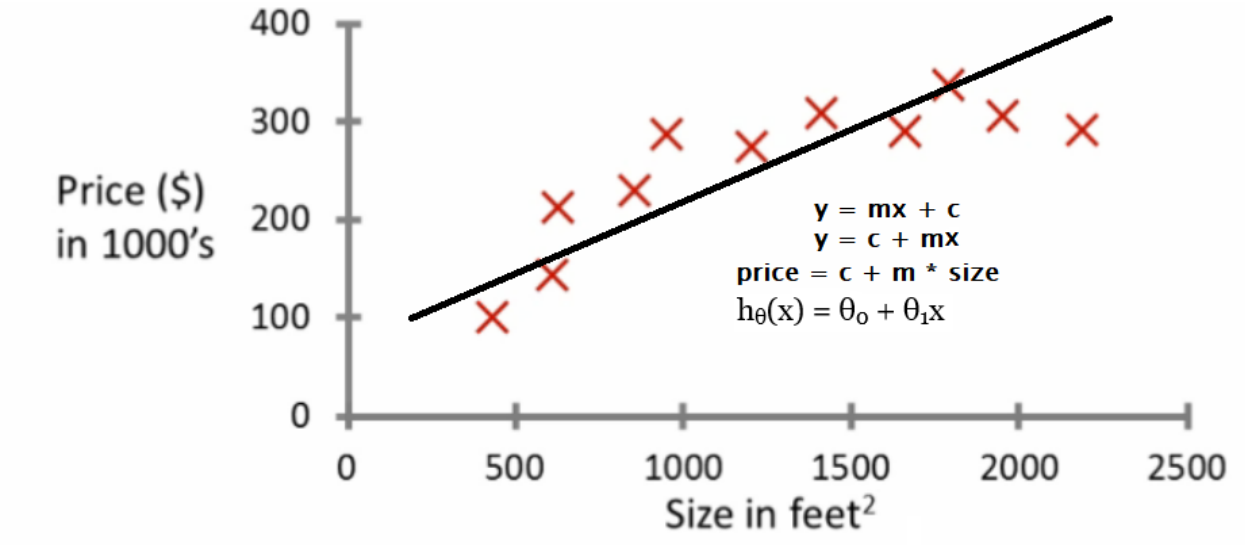
- **Training, testing, and Generalization**

- **Machine learning pipeline:**



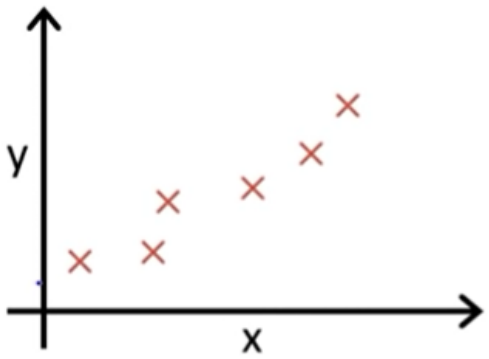
How is the model/hypothesis/function represented for linear regression with one variable:

- Hypothesis: h = b + wx



How to fit the best line/model/hypothesis?

- **Cost function**: Provide the best-fit model
  - = > Mean square error between $\mathbf{f(x^{(i)})}$ and $\mathbf{y^{(i)}}$ is minimum.



Cost Function:

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

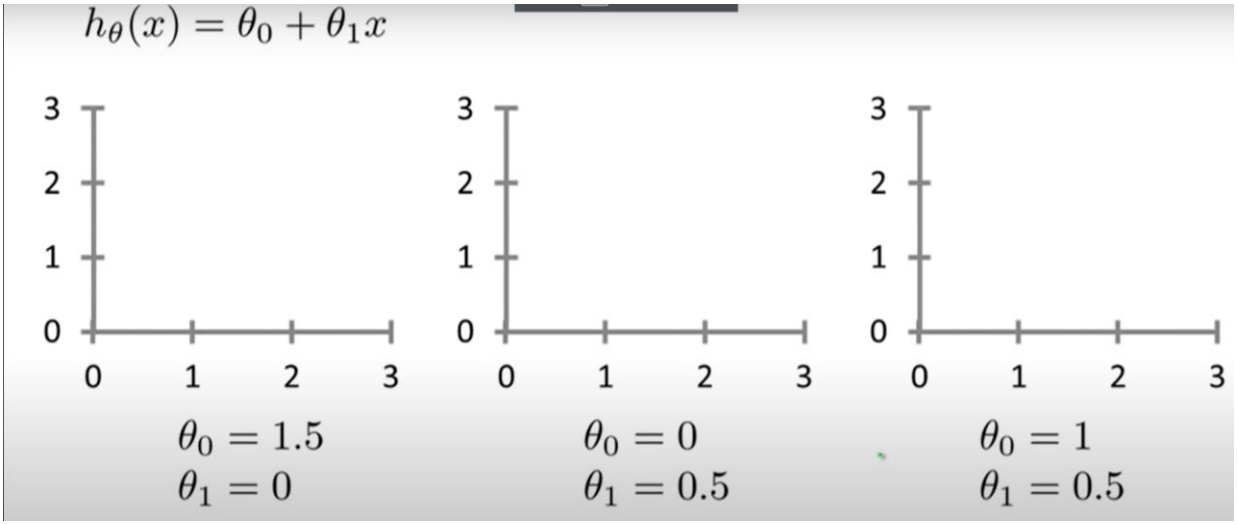
- **Take those thetas that minimize the cost function J**

2.2: Loss/Cost function: Provide the best-fit model

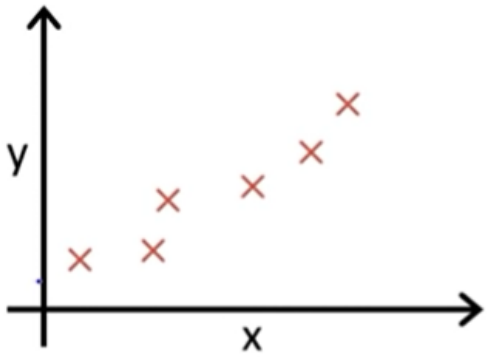
We have:

- Training set:
- Hypothesis: $h_\Theta(x) = \Theta_0 + \Theta_1 x$
- Parameters (Θ_0, Θ_1): need to learn them during the training process

- Question: **How to fit the best line?:**
  - Different values of (Θ_0, Θ_1) give us different hypotheses/functions/ models



- **Answer:** We have to come up with values of parameters (Θ_0, Θ_1) such that model is a good fit for training data
  - Idea: choose theta's so that model output is close to **y** for our training examples (x,y)
  - = > error between $h_\Theta(x)$ and **y** is minimum.



- **To formalize this.**
  - We want to want to solve a **minimization problem**
  - Minimize $(h_\theta(x^{(i)}) - y^{(i)})^2$
    - i.e. minimize the difference between h(x) and y for each example
  - Sum this over the training set
- Cost function: least square error is the most commonly used function for linear regression

Cost Function:

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

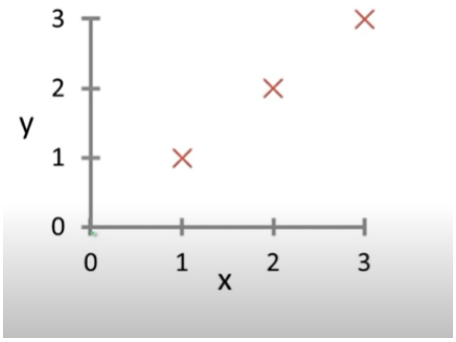
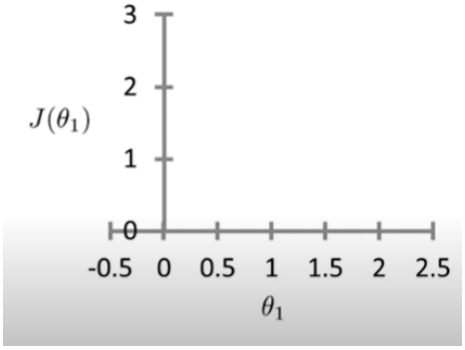
- Find Θ_0 and Θ_1, so that the average 1/2m times the sum of squares between predictions on the training set and actual values of a house on the training set is minimized.
- It is called cost/ error/ objective function.
- It is also called the least square error

2.3: Cost function - a deeper look (Intuition- I)

- Hypothesis: $h_\theta(x^{(i)}) = \Theta_0 + \Theta_1 x^{(i)}$
- Parameters: Θ_0, Θ_1
- Cost $J(\Theta_0, \Theta_1) = 1/2m \sum (h_\theta(x^{(i)}) - y^{(i)})^2$
- Goal is to Minimized $J(\Theta_0, \Theta_1) = 1/2m \sum ((\Theta_0 + \Theta_1 x^{(i)}) - y^{(i)})^2$

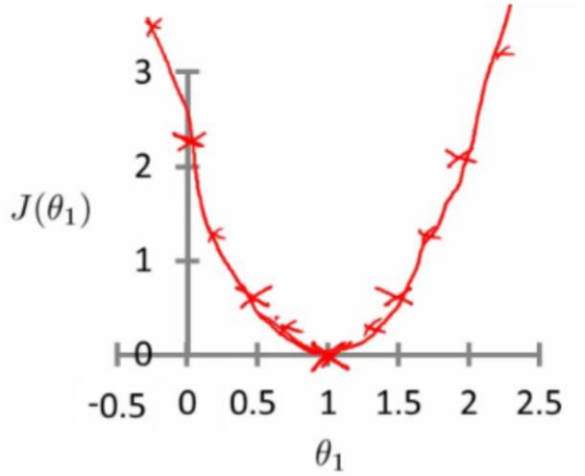
- Simplified hypothesis to make visualizing of cost function J() a bit easier
  - Assumes $\boldsymbol{\theta_0 = 0}$
  - So hypothesis pass through (0,0) (intercept is zero)
  - Hypothesis: $\boldsymbol{h_\theta(x^{(i)}) = \Theta_1 x^{(i)}}$ (just consider the slope)
  - Parameters: $\boldsymbol{\Theta_1}$
  - cost $\boldsymbol{J(\Theta_1) = 1/2m \sum (h_\theta(x) - y)^2}$
  - Goal is to Minimized $J(\Theta_1) = 1/2m \sum (h_\theta(x) - y)^2 = 1/2m \sum (\Theta_1 x^{(i)} - y^{(i)})^2$
  - Suppose we have a training dataset:

X	Y
1	1
2	2
3	3

$h_\theta(x) = \Theta_1 x$ for fix value of θ, this is function of x	$J(\Theta_1) = 1/2m \sum (h_\theta(x) - y)^2$ Is function of parameter Θ_1
 Θ_1 = 1, 0.5, and 0 find $J(\Theta_1)$ =?	

- Question: find $J(\Theta_1) = ?$, and also a draw line for $\Theta_1 = 1, 0.5$, and 0.
- Answer for $J(\Theta_1)$
    - 1)
        - $\theta_1 = 1$
        - $J(\theta_1) = 0$
    - 2)
        - $\theta_1 = 0.5$
        - $J(\theta_1) = \sim 0.58$
    - 3)
        - $\theta_1 = 0$
        - $J(\theta_1) = \sim 2.3$

- If we compute a range of values plot
    - $J(\theta_1)$ vs θ_1 we get a polynomial (looks like a quadratic)



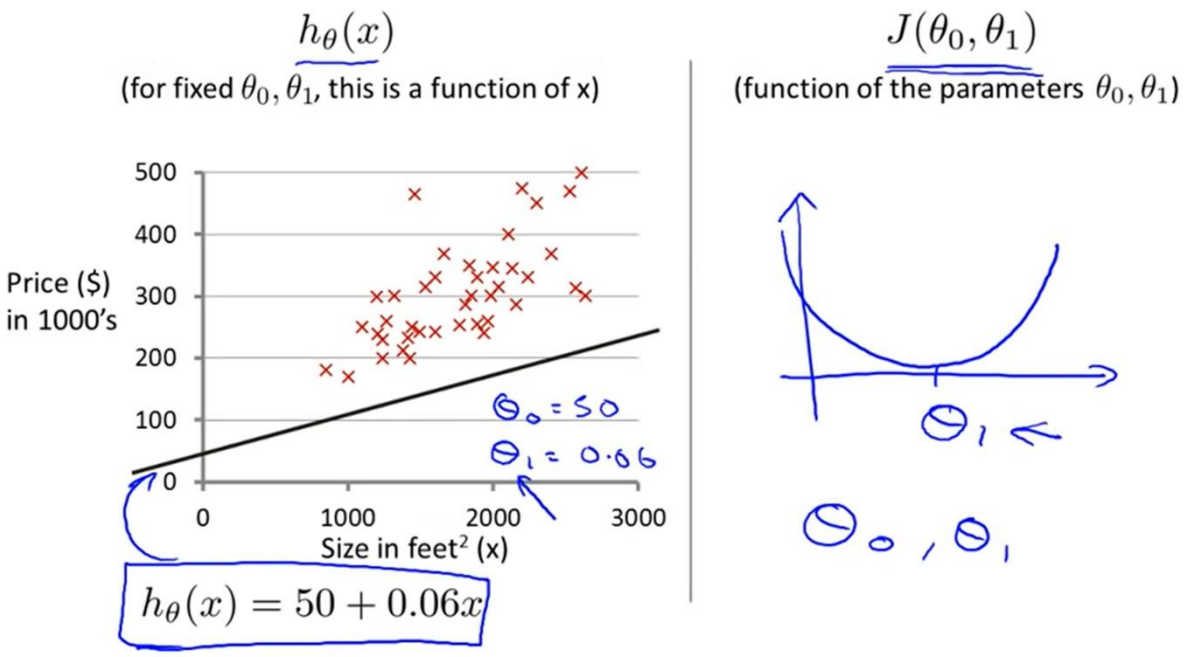
- Conclusion:
- If $\theta_1 = 1$

$J(\theta_1) = 0$ (Best value, it means $\theta_1 = 1$ is the parameter for best fit to training data on the bases of least square error)

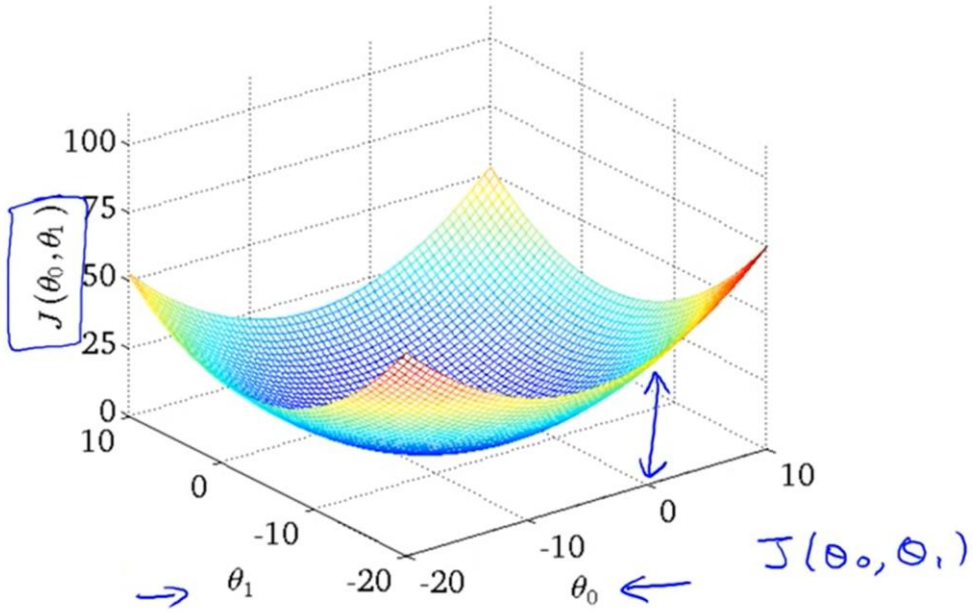
Linear regression with one variable:

Cost function intuition # 2

- Previously we plotted our cost function by plotting
    - θ_1 vs $J(\theta_1)$
- Now for our linear regression problem with one variable we have:
- Hypothesis: $h_\theta(x^{(i)}) = \Theta_0 + \Theta_1 x^{(i)}$
- Parameters: Θ_0, Θ_1
- Cost $J(\Theta_0, \Theta_1) = 1/2m \sum (f_{w,b}(x^{(i)}) - y^{(i)})^2$
- Goal is to Minimized $J(b, w) = 1/2m \sum (((b + wx^{(i)}) - y^{(i)})^2$

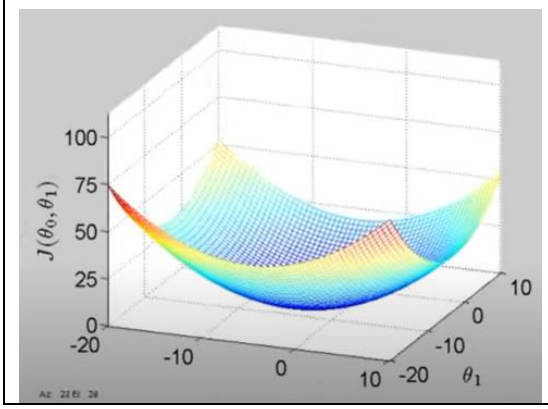


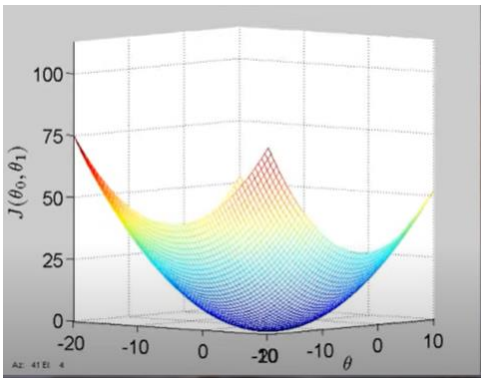
- For a given value of the theta above hypothesis is not the best fit to data.
- Now we what to see error J(…)
- For two parameter the least square error (J(..)) function is still bowl shape function:



- Consider different value of θ_1 and θ_0, we got the different value of $J(\theta_0,\theta_1)$
- Height (Y) of the graph shows the value of value of $J(\theta_0,\theta_1)$
- Height (Y) indicates the value of the cost function, so find where Y is at a minimum

3D visualization of bowl shaped curve:





- Generates a 3D surface plot where the axis are
  - X = w = 0.06
  - Z = b = 10
  - Y = J(b,w) $= 1/2m \sum((f(x^{(i)}) - y^{(i)})^2) = 1/2m \sum((b + wx^{(i)} - y^{(i)})^2)$

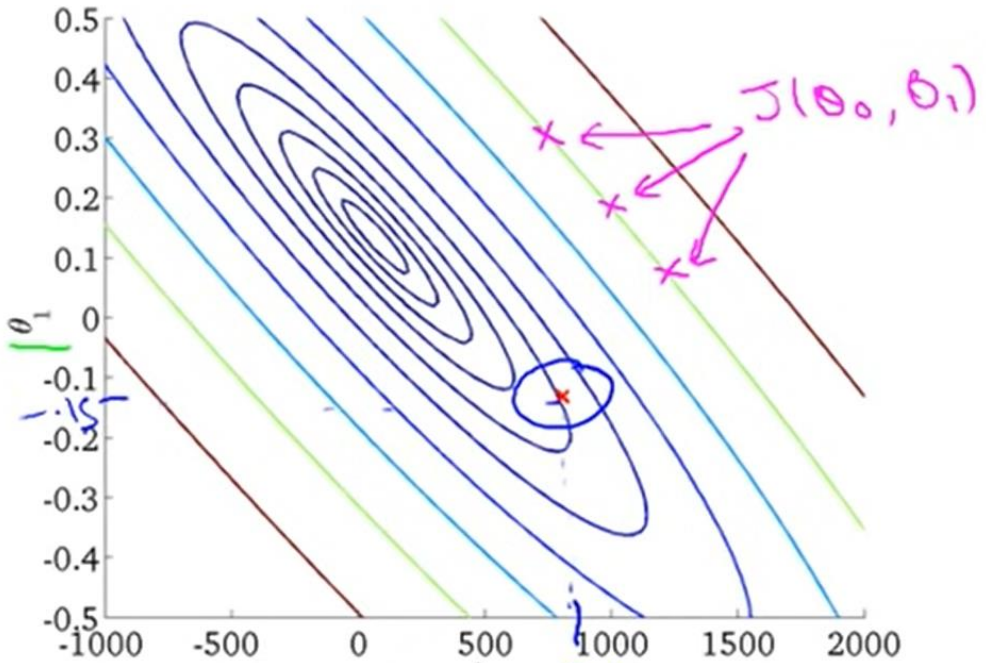
- Plot to show Error value (J()) becomes a bit more complicated to visualize

Countour Figures/ contour plot:

- It is a convenient way to visualize the cost function.
- Every ring/ellipse of specific color has the same height, which means they have the same **error value** ($J(\theta_0,\theta1)$)
- Bottom of the bowl shape is represented in the middle
- Assume the see the bowl shape at the top
- Consider different values of thetas and find the error and ultimately try to find the best fit.
  - Consider the specific value of $J(\theta_0,\theta1)$, find its values from the contour figure, and then draw a line on the training data to see if it fit or not

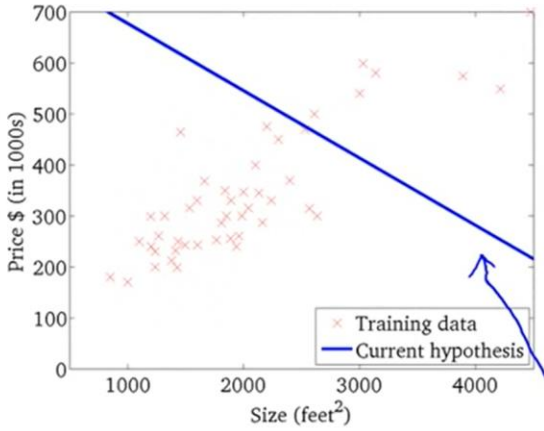
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



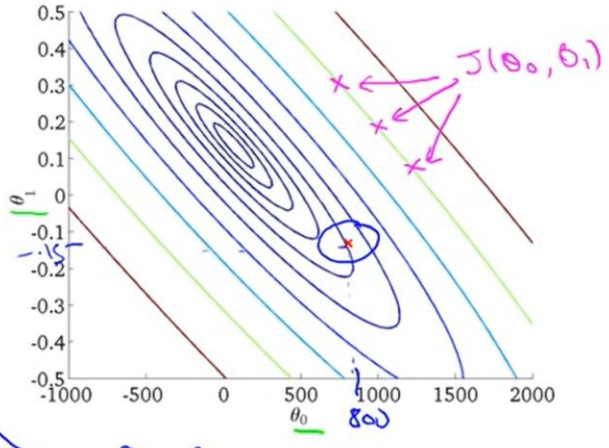
$h_\theta(x)$

(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)

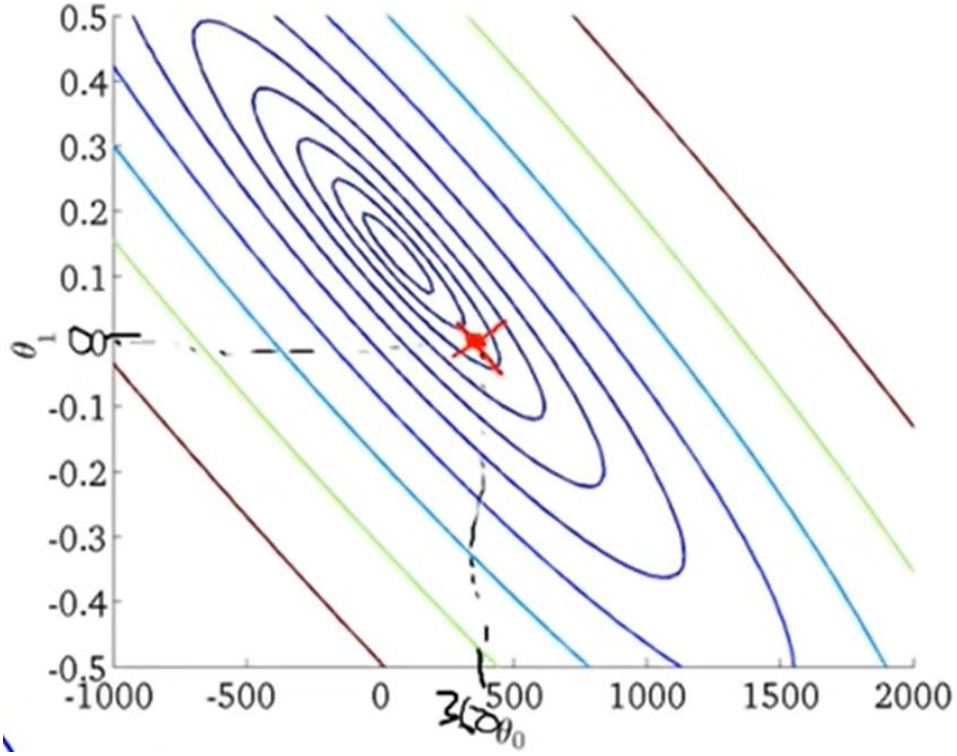


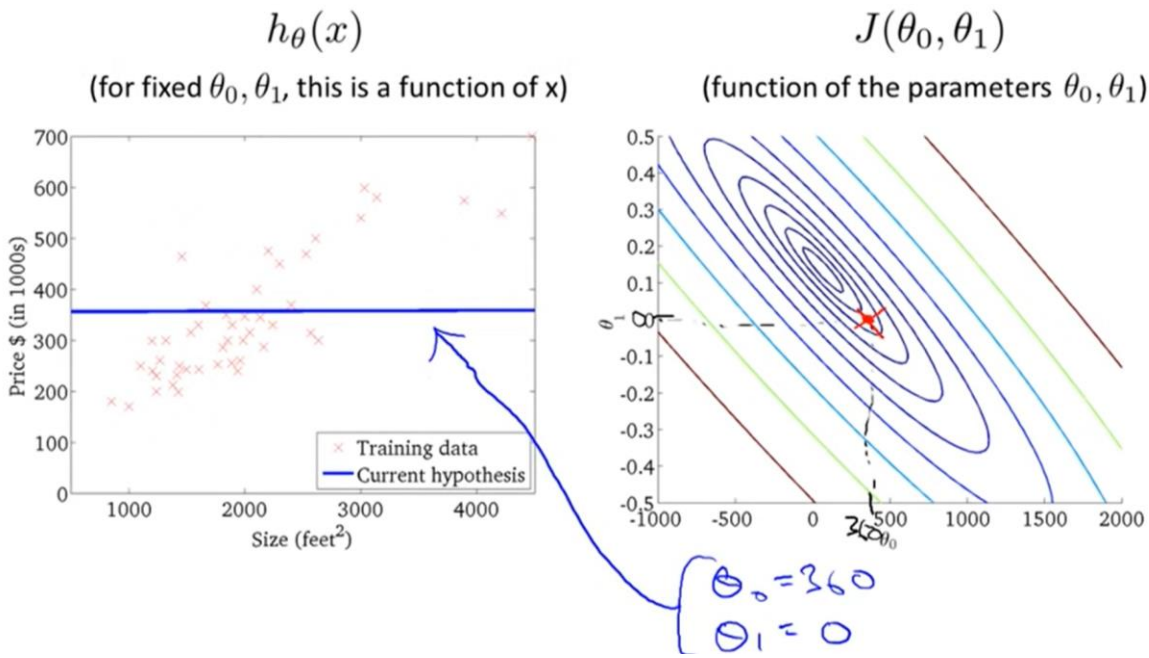
θ_0, θ_1

As the shown line is not well fitted to the data, so the corresponding cost is high

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



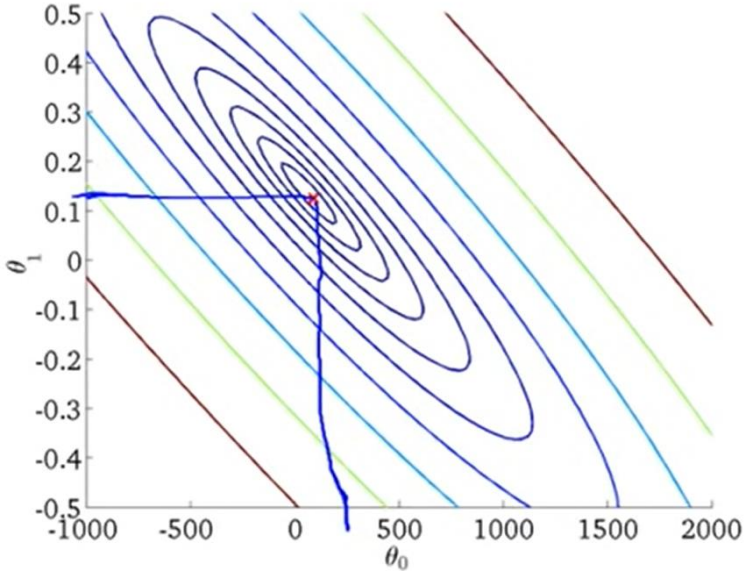


- $h_\theta(x^{(i)}) = b + wx^{(i)} = 360 + 0\ x^{(i)}$

- Model is still not a best fit

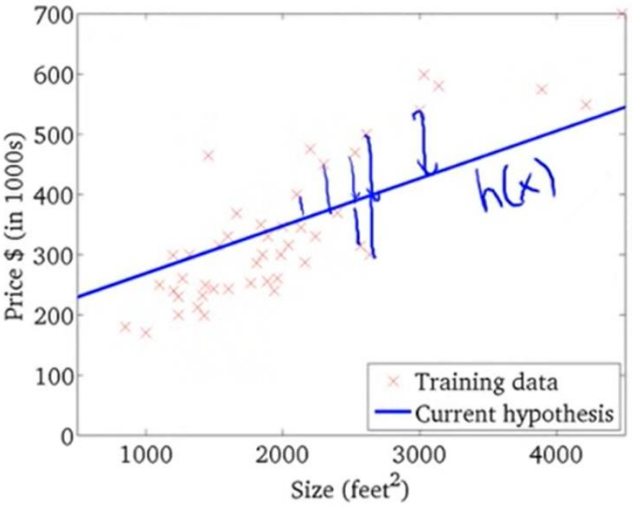
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



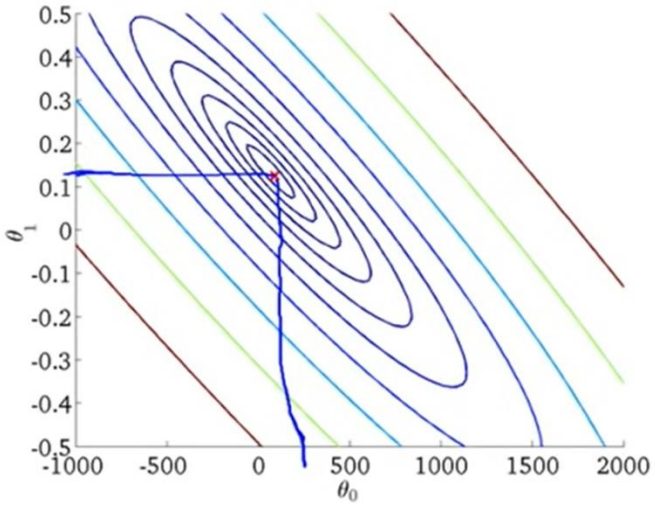
$h_\theta(x)$

(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)



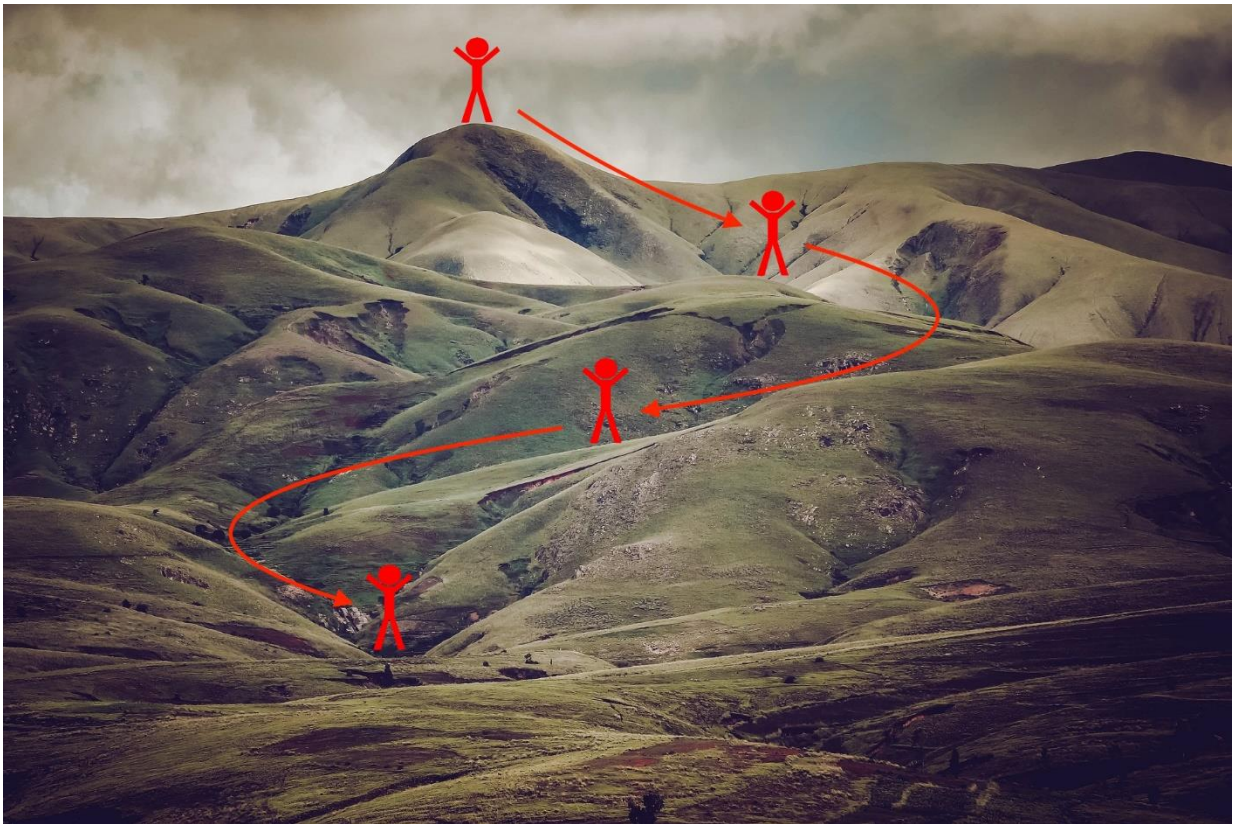
- J(b,w) = 1/2m $\sum$((f($x^{(i)}$) - $y^{(i)}$)^2) = 1/2m $\sum$((b + w$x^{(i)}$)- $y^{(i)}$)^2)

- This gives a better hypothesis, but still not great - not in the center of the contour plot
- least square error is minimum at this point
- Finally, we find the minimum, which gives the best hypothesis
- **It means that the means square error of predicted values and actual y values corresponding to given inputs is minimum for this hypothesis**

- Doing this by eye/hand is not an efficient value
  - What we really want is an **automatic** and efficient algorithm to **find b and w** to minimum cost function => gradient descent do it for us
  - Secondly for **higher dimensional features** it is even more difficult to visualize the relationship between thetas, cost function, and hypothesis but still gradient descent is capable enough to find the thetas that minimized the cost function J()and provides us the best fit hypothesis.

Gradient descent:

- It is used for minmization of different loss function that are used for different problem. Here we discussed it for any arbitrary cost function and then see its used for linear regression prolem.



- **Problem formulation:**
  - We have an error function, **J(b, w),** for linear regression with one variable
  - We want to get **min J(b, w)** automatically

- Gradient descent algorithm automatically finds **b** and **w** corresponding to that we have min $J(\theta_0, \theta_1)$
- Gradient descent applies to more **general functions**

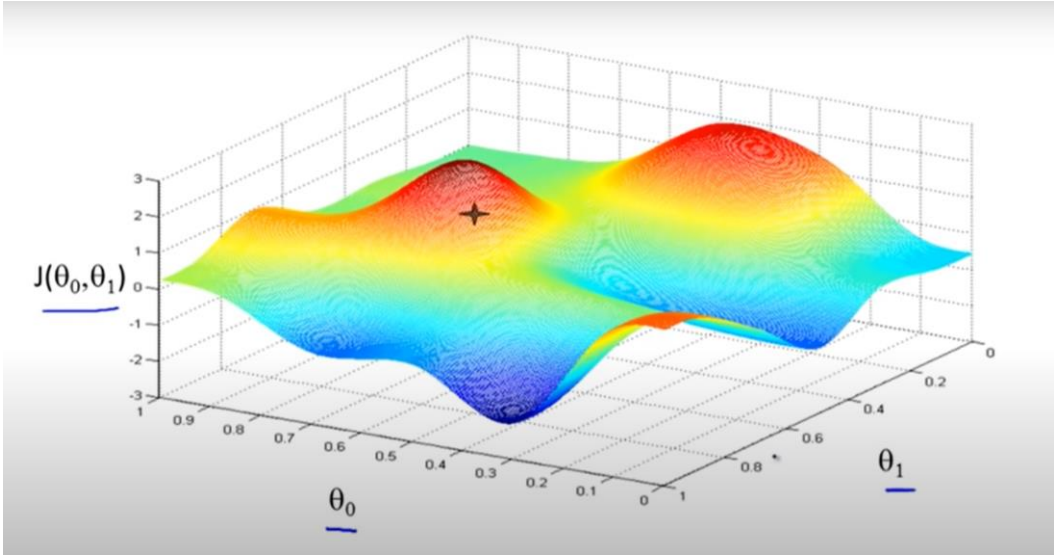
  - $J(b, w_1, w_2 \ldots. w_n)$
  - min $J(b, w_1, w_2 \ldots. w_n)$

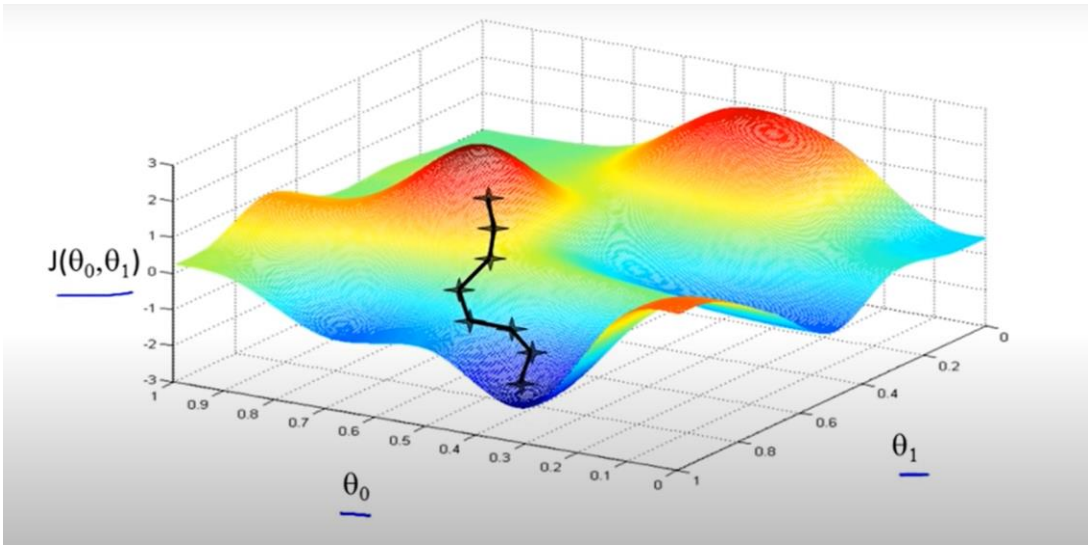
- **How does it work?**

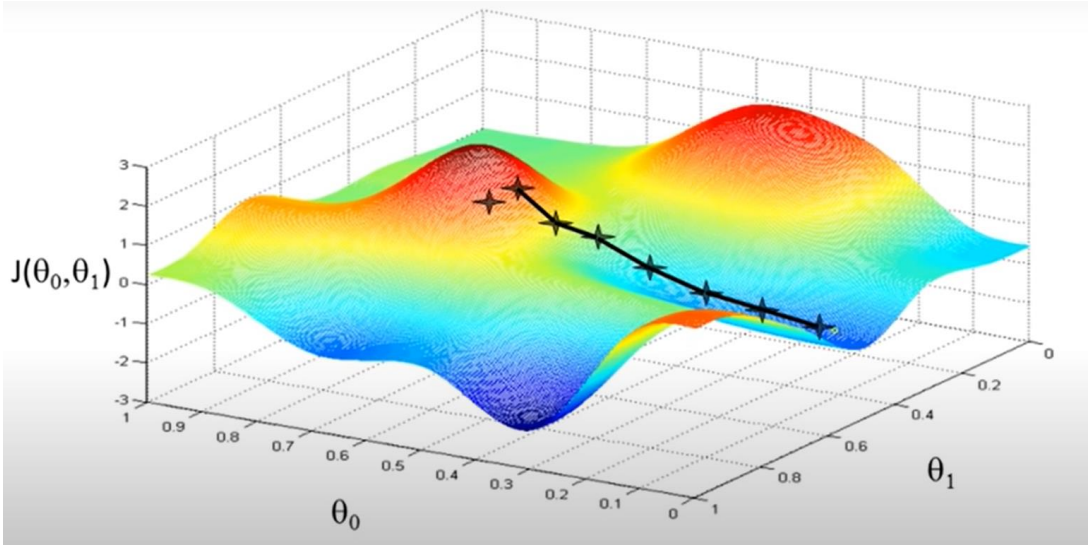
    1. Start with some value of b and w
    2. Keeping changing b and w a little bit to reduce J(b,w)
        - Each time to change the parameters, select the gradient (steepest direction) which reduces J(b,w) the most possible
    3. Repeat, do so until you converge to a minimum

- **Graphical representation:**

https://www.mygreatlearning.com/blog/gradient-descent/ [see more explanation]







- Gradient descent has an interesting property
- Here we can see one initialization point led to one local minimum
- The other led to a different one
- **In our case of linear regression with one variable, we have the cost J() function that has single minima that are also called global minima of the function**

Mathematical representation: A more formal definition

- Do the following until convergence

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

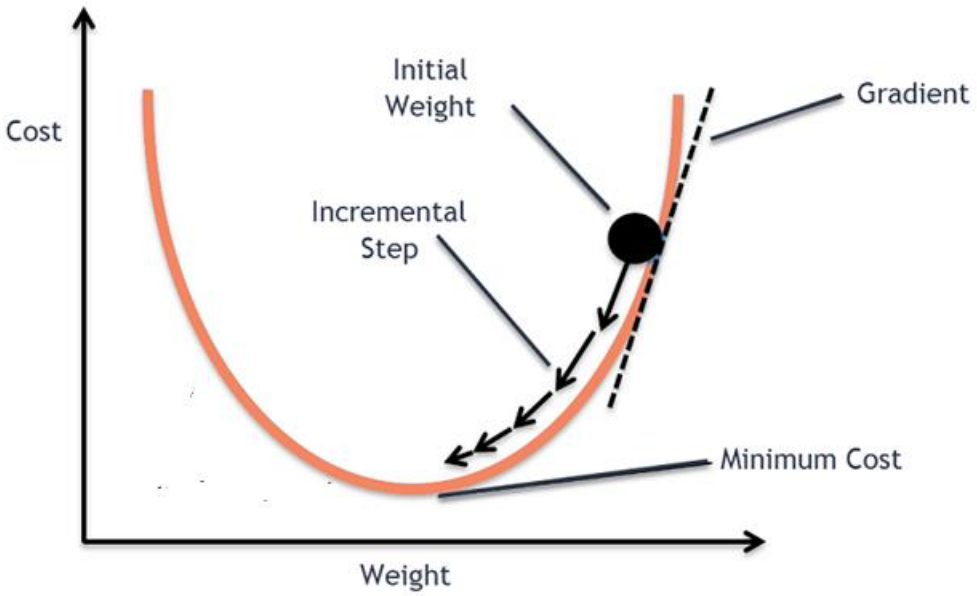
- What does this all mean?
  - Update θ_j by setting it to (θ_j - α) times the partial derivative of the cost function with respect to θ_j
- **Terms**
  - :=
    - Denotes assignment
  - α (alpha)
    - Is a number called the **learning rate**
    - Controls how big a step you take
      - If α is big have an aggressive gradient descent
      - If α is small take tiny steps
  - derivative term: slop of a line tangent to the function

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

Understanding the Gradient descent for your problem

- To understand gradient descent, we'll return to a simpler function where we minimize one parameter to help explain the algorithm in more detail
  - min $J(\theta_1)$ where θ_1 is a real number
- **Two key terms in the algorithm**
  - alpha
  - derivative term
- **Derivative term:** What is the slope of a line that is tangent to the function J at a point
  - **Slope is height/ width**

$$\frac{\partial}{\partial \theta_j} J(\theta_1)$$



- Derivative says
  - Let's take the tangent at the point and look at the slope of the line
  - So moving towards the minimum (down) will get a derivative (a positive value), alpha is always positive, so will update $j(\theta_1)$ to a smaller value
  - Similarly, if we're moving a slope we make $j(\theta_1)$ a bigger numbers
  - If the derivative a positive number ??
  - If derivate is a negative number??

- **When already at a local minimum one step does nothing**
    - Gradient of tangent/derivative is 0
    - So derivative term = 0
    - alpha * 0 = 0
    - So $\theta_1 = \theta_1 - 0$
    - So θ_1 remains the same

Alpha term (α)

- What happens if alpha is too small or too large
    - Too small
        - Take baby steps
        - Takes too long
    - Too large
        - Can overshoot the minimum and fail to converge

- As you approach the global minimum the derivative term gets smaller, so your update gets smaller, even with **alpha is fixed**
    - Means as the algorithm runs you take smaller steps as you approach the minimum due to **smaller derivate /slope values**
    - So no need to change alpha over time

3.7 Put together **Gradient Descent for Linear Regression with one variable:**

- Apply the gradient descent to minimize the square error function of linear regression
- Need to find the partial derivative of the error function with respect to theta0 and theta 1
- Gradient descent with cost function a (least square error a is a convex function that gives us a global minimum) gives us an algorithm for linear regression for fitting the straight line in given data.

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

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

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

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

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

θ_0 $j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) =$

θ_1 $j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) =$

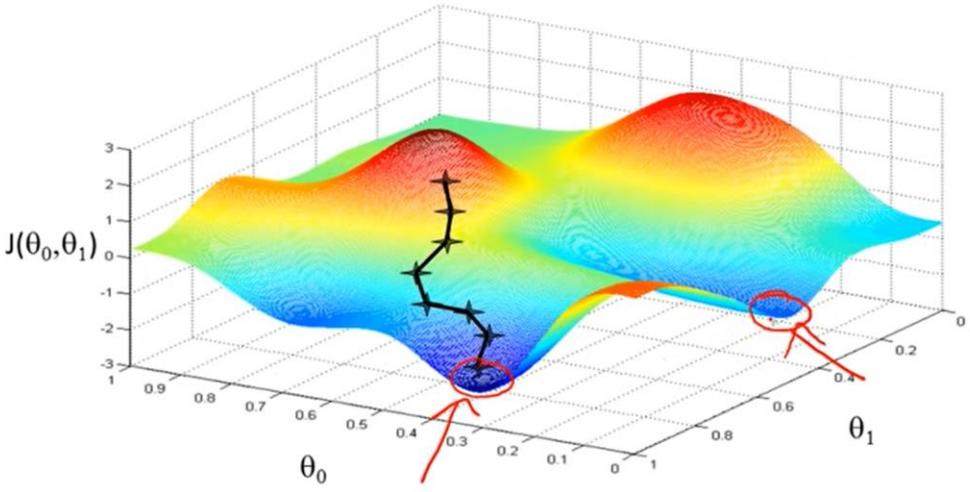
Gradient descent algorithm

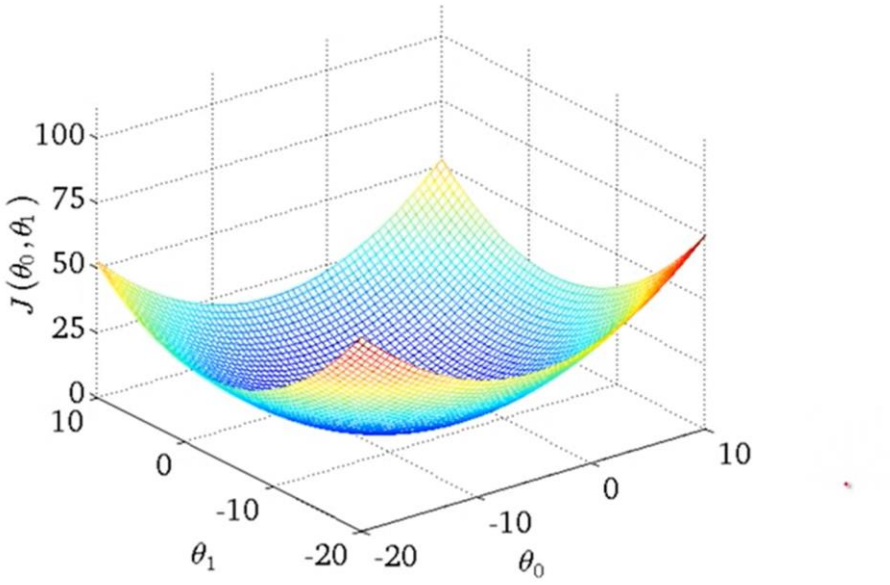
repeat until convergence {

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

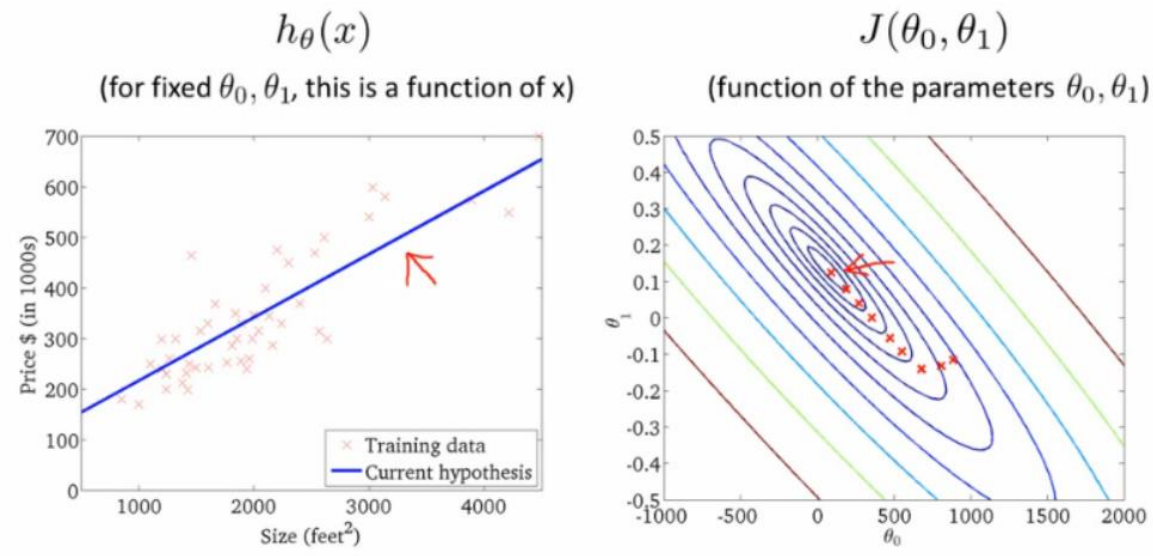
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

}





- Bowl shape function for linear regression is a **convex function** that has global minima. We has seen that LSE is a bowl shape
- In action
  - Initialize values to
    - $\theta_0 = 900$
    - $\theta_1 = -0.1$



Batch gradient descent:

- This is actually Batch Gradient Descent
  - Each step of gradient descent using all the training data
    - Each step compute over m training examples
  - Sometimes non-batch versions exist, which look at small data subsets
    - We'll look at other forms of gradient descent (to use when m is too large) later in the course

Next to do: Important extensions

- Normal equation for numeric solution
- We can learn with a larger number of features
- Review of linear algebra