

## Week 2

July 10, 2021


## Linear Regression

The previously described home price data example is an example of a supervised learning regression problem.

### Notation (used throughout the course)

- $m$  = number of training examples.
- $x$ 's = input variables / features.
- $y$ 's = output variable "target" variables.
- $(x, y)$  - single training example.
- $(x^i, y^j)$  - specific example (ith training example).

Size in feet <sup>2</sup> (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...



### With our training set defined - how do we use it?

- Take training set.
- Pass into a learning algorithm.
- Algorithm outputs a function ( $h$  = hypothesis).
- This function takes an input (e.g. size of new house).
- Tries to output the estimated value of  $Y$ .

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

- $Y$  is a linear function of  $x$ !
- $\theta_0$  is zero condition.
- $\theta_1$  is gradient.

## Cost function

- We may use a cost function to determine how to fit the best straight line to our data.
- We want to solve a minimization problem. Minimize  $(h_{\theta}(x) - y)^2$ .
- Sum this over the training set.

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

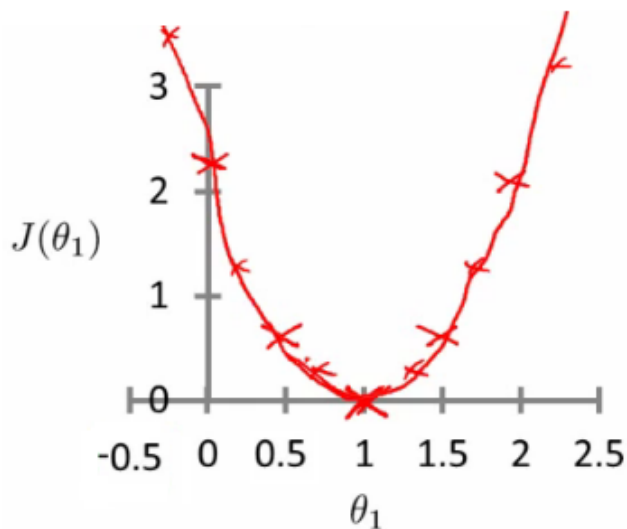
## Example

For  $\theta_0 = 0$  we have:

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

Data:

- $\theta_1 = 1$  and  $J(\theta_1) = 0$ .
- $\theta_1 = 0.5$  and  $J(\theta_1) = 0.58$ .
- $\theta_1 = 0$  and  $J(\theta_1) = 2.3$ .



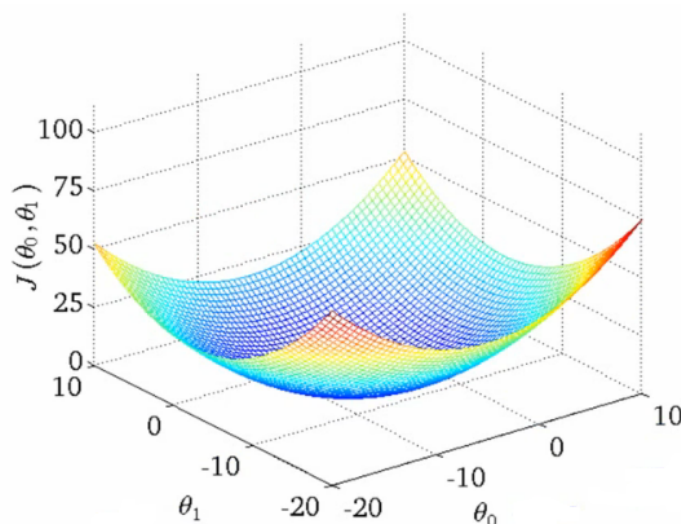
The optimization objective for the learning algorithm is find the value of  $\theta_1$  which minimizes  $J(\theta_1)$ . So, here  $\theta_1 = 1$  is the best value for  $\theta_1$ .

## A deeper insight into the cost function - simplified cost function

The real cost function takes two variables as parameters!  $J(\theta_0, \theta_1)$ .

We can now generate a 3D surface plot where axis are:

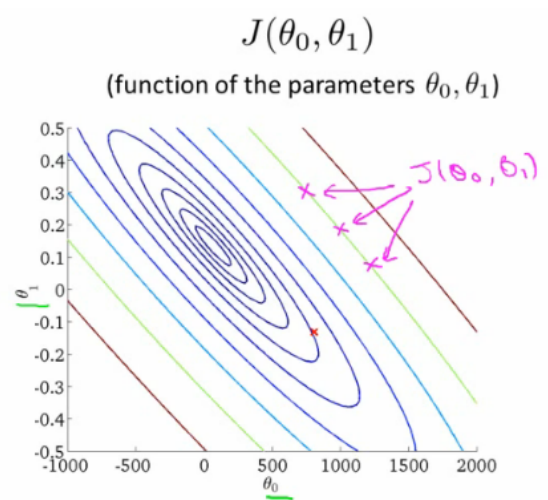
- $X = \theta_1$ .
- $Z = \theta_0$ .
- $Y = J(\theta_0, \theta_1)$ .



The best hypothesis is at the bottom of the bowl.

Instead of a surface plot we can use a contour figures/plots.

- Set of ellipses in different colors.
- Each colour is the same value of  $J(\theta_0, \theta_1)$ , but obviously plot to different locations because  $\theta_1$  and  $\theta_0$  will vary.
- Imagine a bowl shape function coming out of the screen so the middle is the concentric circles.

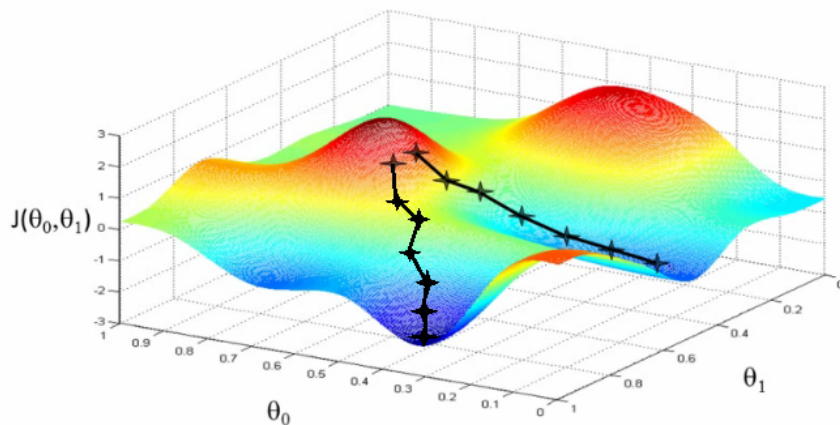


The best hypothesis is located in the center of the contour plot.

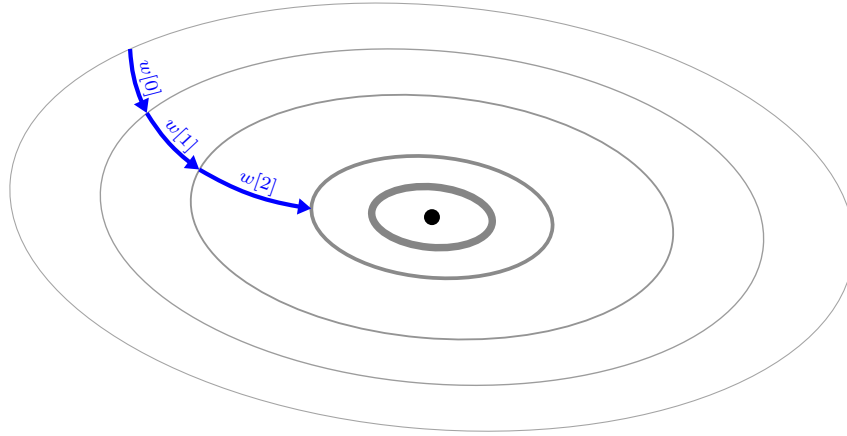
## Gradient descent algorithm

- Begin with initial guesses, could be (0,0) or any other value.
- Continually change values of  $\theta_0$  and  $\theta_1$  to try to reduce  $J(\theta_0, \theta_1)$ .
- Continue until you reach a local minimum.
- Reached minimum is determined by the starting point.

Surface plot of gradient descent.



### Contour plot of gradient descent.



---

**Algorithm 1** Gradient Descent

---

```
1: while not converged do
2:    $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1).$ 
3:   (for  $j = 1$  and  $j = 0$ )
```

---

### Two key terms in the algorithm:

- $\alpha$  term
- Derivative term

### Partial derivative vs. derivative

- Use partial derivative when we have multiple variables but only derive with respect to one.
- Use derivative when we are deriving with respect to all the variables.

### Derivative says:

- Let's look at the slope of the line by taking the tangent at the point.
- As a result, going towards the minimum (down) will result in a negative derivative; nevertheless, alpha is always positive, thus  $J(\theta_1)$  will be updated to a lower value.
- Similarly, as we progress up a slope, we increase the value of  $J(\theta_1)$ .

### $\alpha$ term

- If it's too small, it takes too long to converge.
- If it is too large, it may exceed the minimum and fail to converge.

### When you get to a local minimum

- Gradient of tangent/derivative is 0
- So derivative term = 0
- $\alpha \cdot 0 = 0$
- So  $\theta_1 = \theta_1 - 0$ .
- So  $\theta_1$  remains the same.

## Linear regression with gradient descent

Apply gradient descent to minimize the squared error cost function  $J(\theta_0, \theta_1)$ .

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2\end{aligned}$$

For each case, we must determine the partial derivative:

$$\begin{aligned}j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_j} \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \\ j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_j} \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)}\end{aligned}$$

The linear regression cost function is always a convex function - always has a single minimum. So gradient descent will always converge to global optima.

## Two extension to the algorithm

### Normal equation for numeric solution

- To solve the minimization problem we can solve it  $[\min J(\theta_0, \theta_1)]$  exactly using a numeric method which avoids the iterative approach used by gradient descent.

- Normal equations method.
- Can be much faster for some problems, but it much more complicated (will be covered in detail later).

### **We can learn with a larger number of features**

- e.g. with houses, Size, Age, Number bedrooms, Number floors...
- Can't really plot in more than 3 dimensions.
- Best way to get around with this is the notation of linear algebra (matrices and vectors).