Data Science & ML Course

Lesson #18 Linear Regression

Ivanovitch Silva

November, 2018

- We are going to start by covering **linear regression**
  - One variable
  - Multiples variables
- We discuss the application of linear regression to **housing price prediction**
- Present the notion of a **cost function**
- Introduce the **gradient descent** method for learning.
- Refresher on **linear algebra concepts**.

Update from repository

```
git clone https://github.com/ivanovitchm/datascience2machinelearning.git
```

Or ….

```
git pull
```

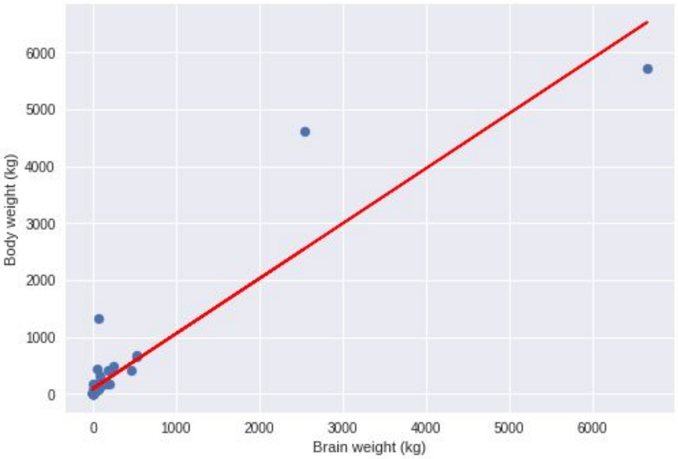
Regression

5 kilograms

200 kilograms

1.5 kilograms

????

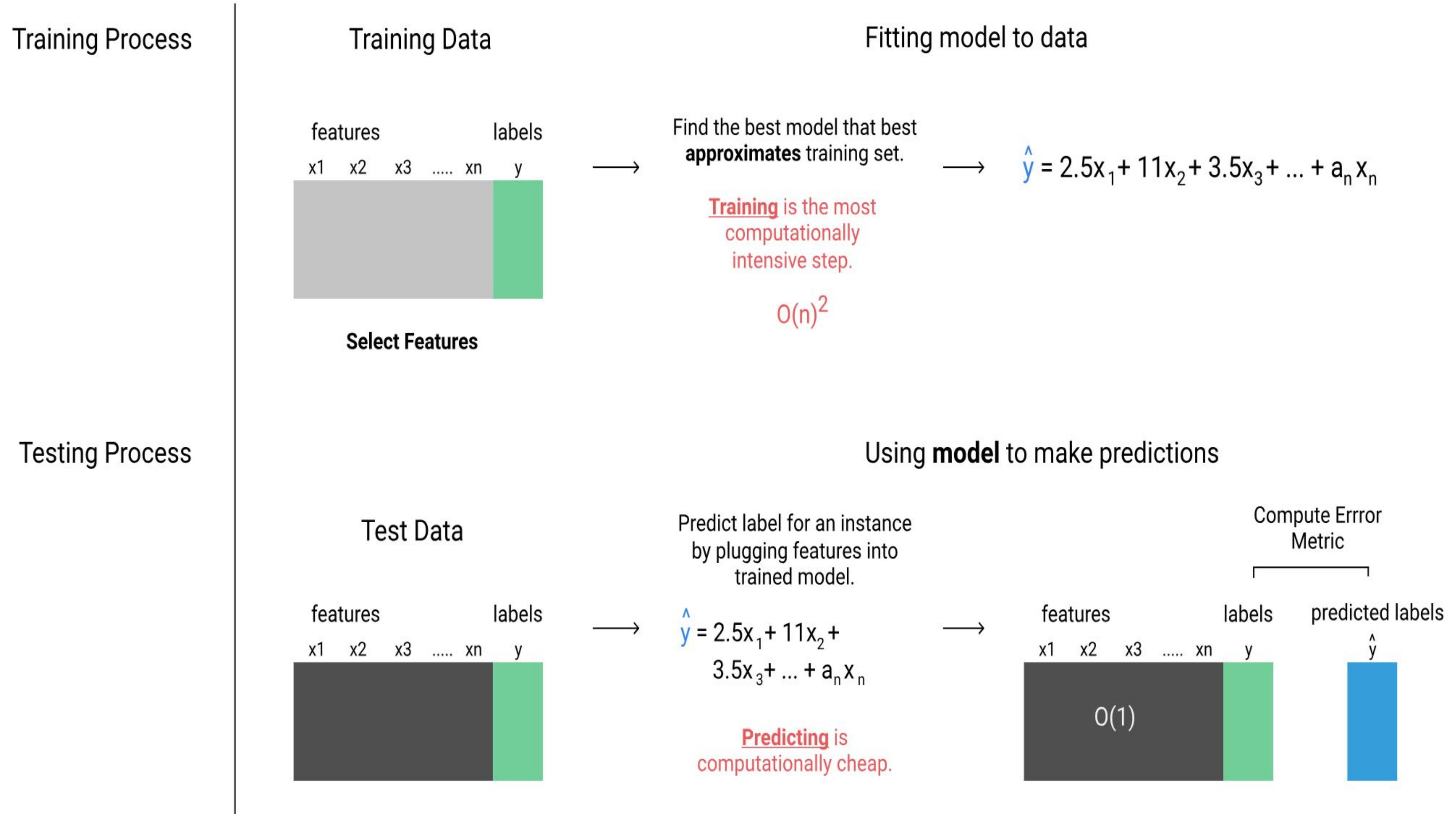



```python
import pandas as pd
from sklearn import linear_model
import matplotlib.pyplot as plt

#read data
df = pd.read_fwf('brain_body.txt')
X = df[['Brain']]
y = df[['Body']]


#train model on data
model = linear_model.LinearRegression()
model.fit(X, y)

#visualize results
plt.scatter(X.values,y.values)
plt.plot(X.values,model.predict(X),color='red')
plt.xlabel('Brain weight (kg)')
plt.ylabel('Body weight (kg)')

plt.show()
```

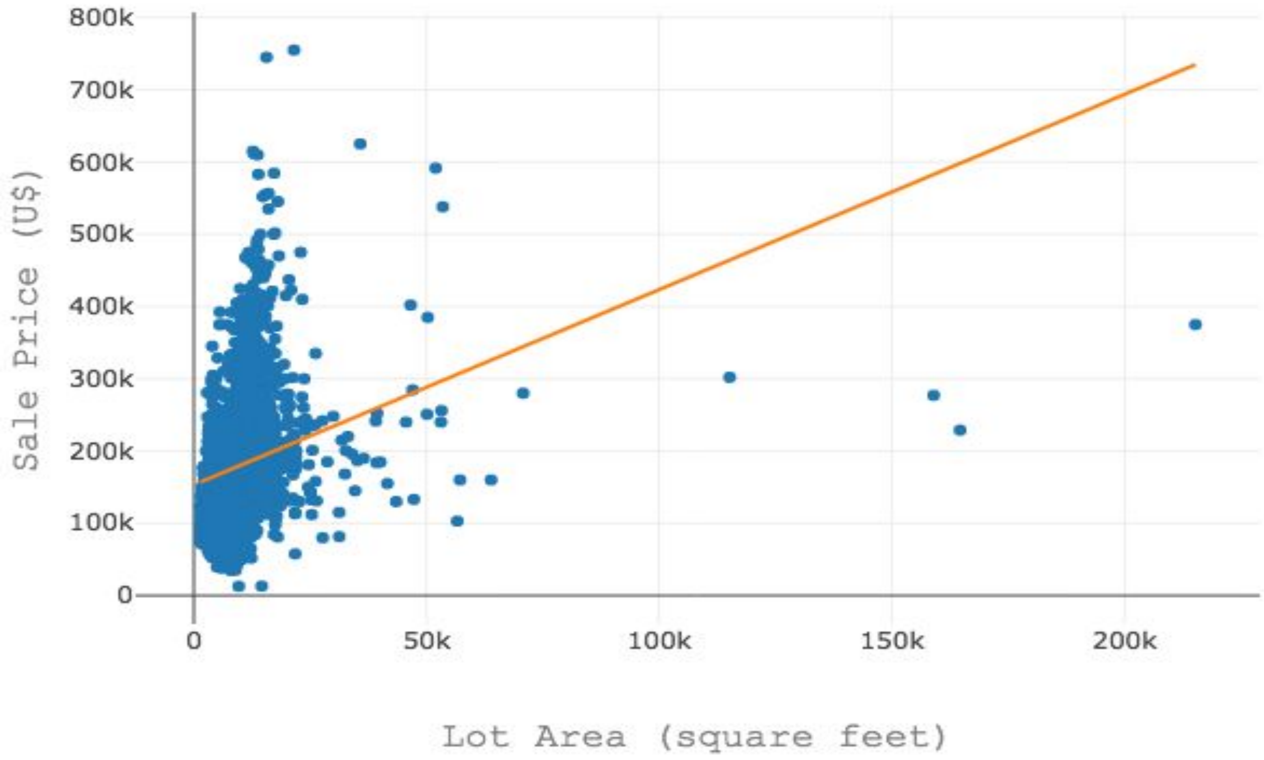
1.5 kilograms

????

1_hello_world.ipynb


Training Process
Training Data
Fitting model to data
features
labels
x1 x2 x3 ..... xn y
Find the best model that best **approximates** training set.
$\hat{y} = 2.5x_1 + 11x_2 + 3.5x_3 + ... + a_n x_n$
Training is the most computationally intensive step.
$O(n)^2$
Select Features
Testing Process
Using **model** to make predictions
Test Data
Predict label for an instance by plugging features into trained model.
Compute Errror Metric
features
labels
x1 x2 x3 ..... xn y
$\hat{y} = 2.5x_1 + 11x_2 + 3.5x_3 + ... + a_n x_n$
features
labels
predicted labels
x1 x2 x3 ..... xn y $\hat{y}$
$O(1)$
Predicting is computationally cheap.

Linear Regression - Housing Price



Lot Area	Overall Qual	Year Built	Yr Sold	SalePrice
31770	6	1960	2010	215000
11622	5	1961	2010	105000
14267	6	1958	2010	172000
11160	7	1968	2010	244000
13830	5	1997	2010	189900

One variable

Multiple variables

Reproduce this fig. using plotly and scikit-learn

Read sections 1 and 2

2. The linear Regression Model.ipynb

Linear Regression with One Variable

Notation:

- m - number of training examples
- X's - input variable/features
- y's - output variable/ target variable

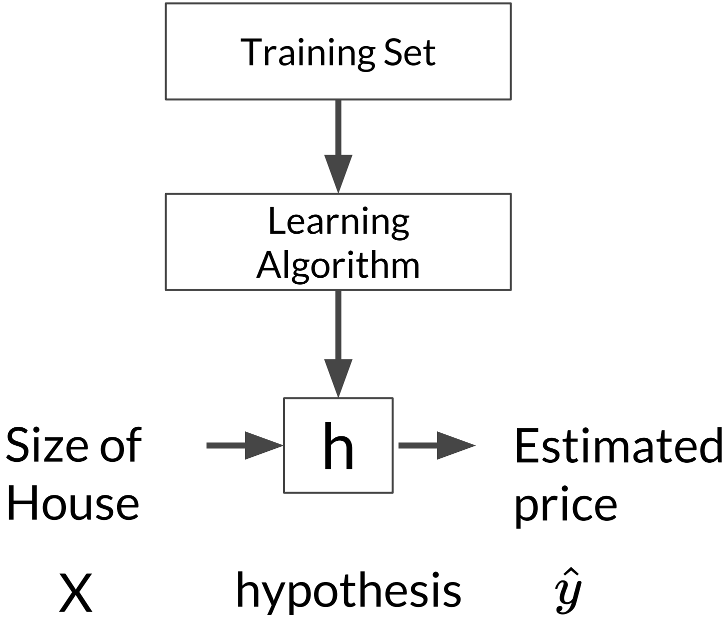
X	y
Lot Area	SalePrice
31770	215000
11622	105000
14267	172000
11160	244000
13830	189900

$m = 1465$

$X^{(1)} = 31770 \quad y^{(1)} = 215000$
$X^{(2)} = 11622 \quad y^{(2)} = 105000$
$X^{(3)} = 14267 \quad y^{(3)} = 172000$

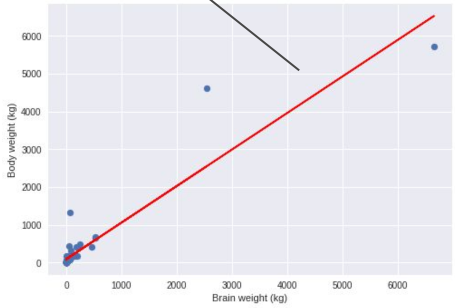
$(X^{(i)}, y^{(i)}) = i^{th}$ training example

Model Representation (linear reg. one variable)

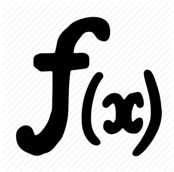


How do we represent h?

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



Cost Function



"minimize the error"

Cost Function (Linear Reg. One Var.)

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

θ_i = parameters

How to choose θ_i ?

Training Set

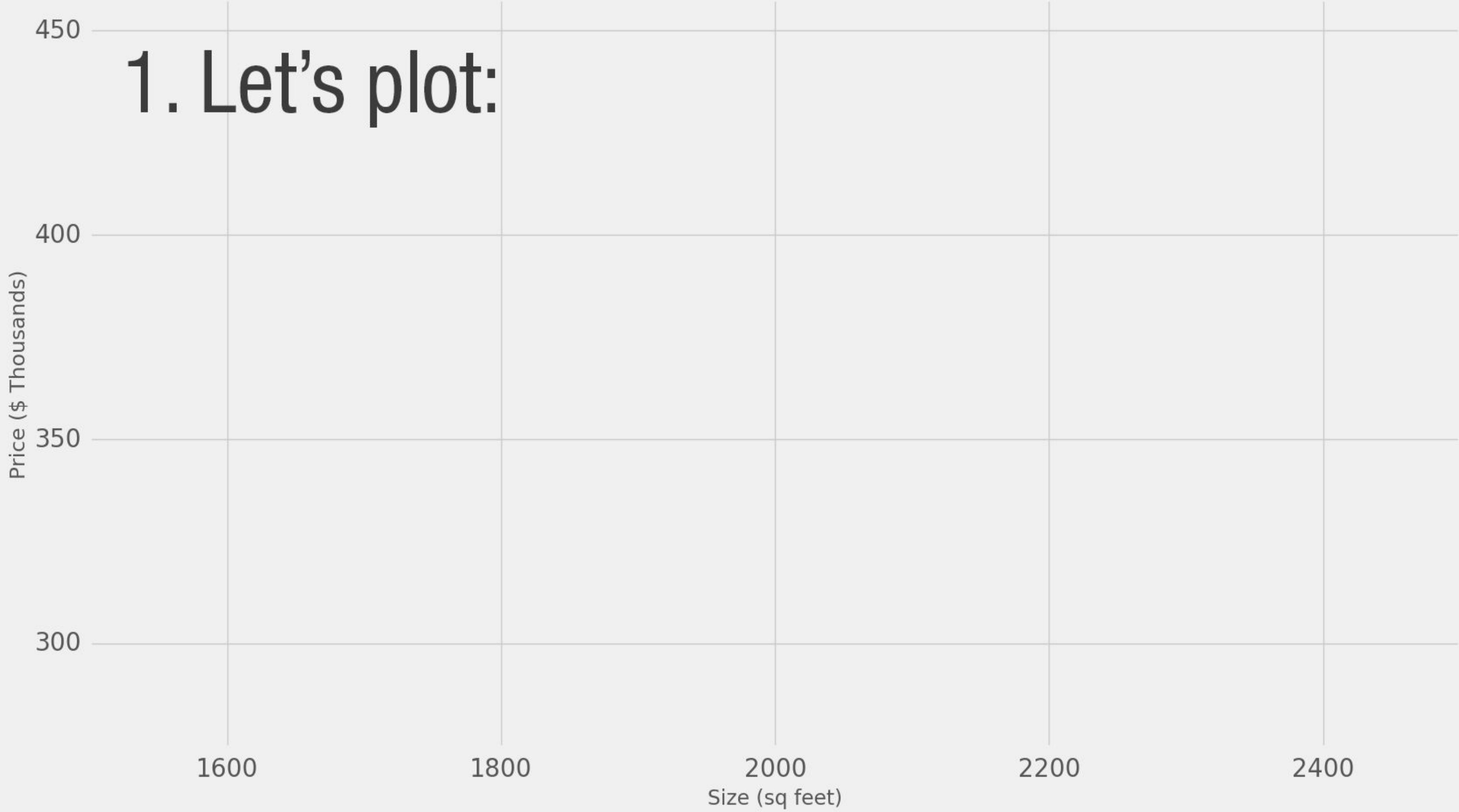
X y

m = 1465

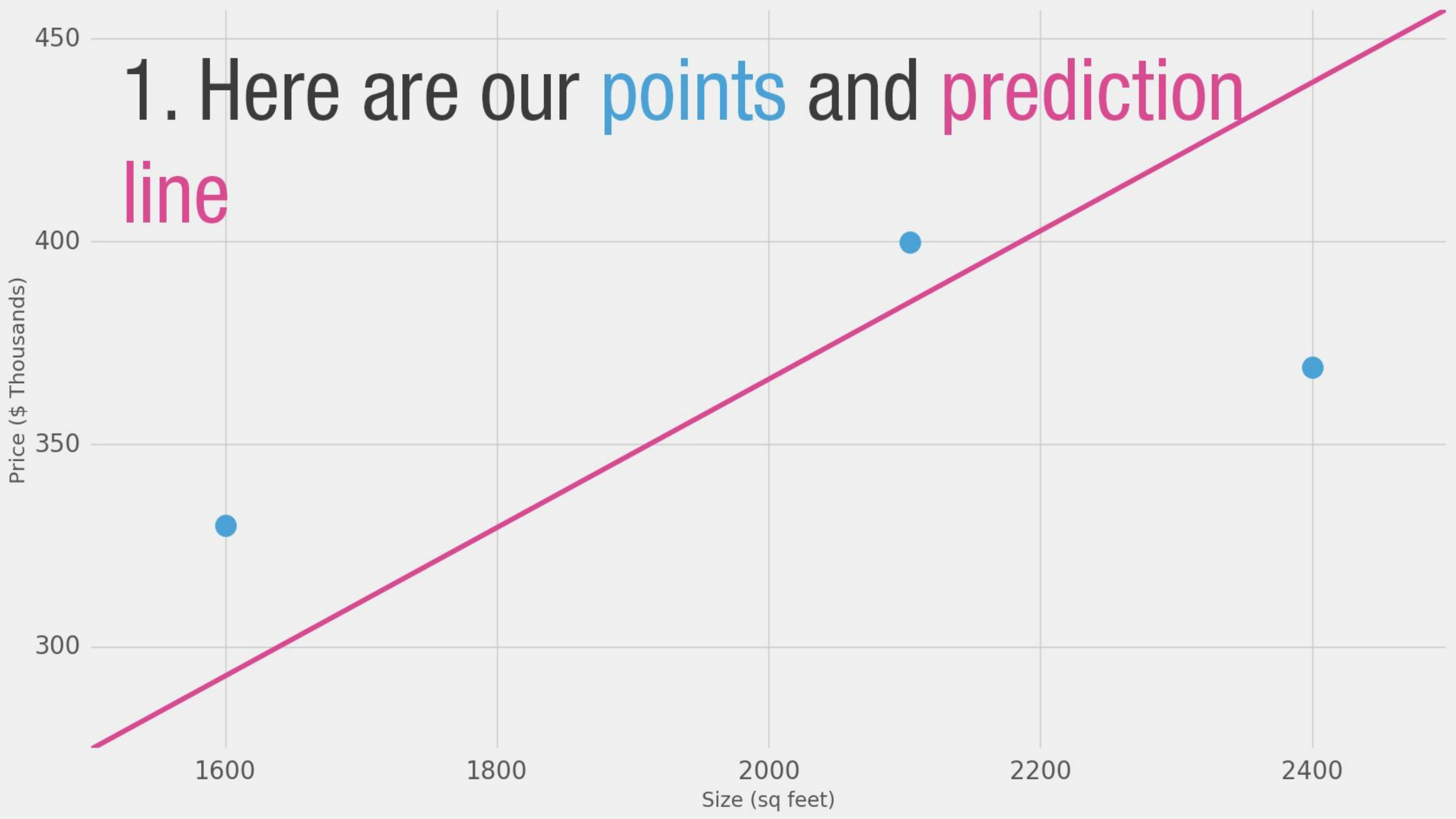
Lot Area	SalePrice
31770	215000
11622	105000
14267	172000
11160	244000
13830	189900

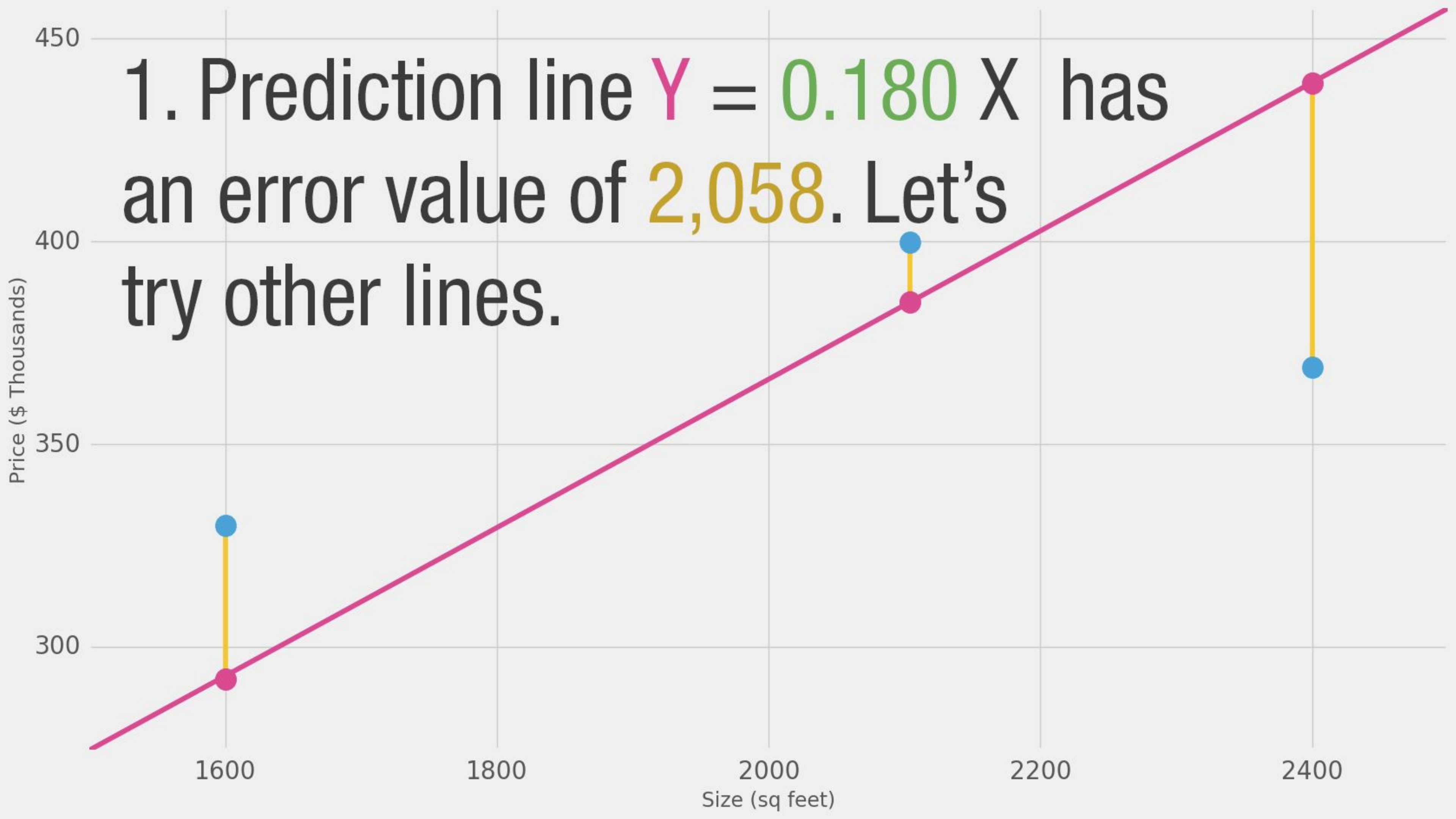
Cost Function Intuition #01 (linear reg. One var)

1. Let’s plot:



450
400
350
300
Price ($ Thousands)
1600
1800
2000
2200
2400
Size (sq feet)


1. Here are our points and prediction line
450
400
350
300
Price ($ Thousands)
1600
1800
2000
2200
2400
Size (sq feet)


1. Prediction line Y = 0.180 X has an error value of 2,058. Let's try other lines.
450
400
350
300
1600
1800
2000
2200
2400
Price ($ Thousands)
Size (sq feet)

Cost Function (square error function)

Training Set (m instances)

	Lot Area	SalePrice	
$x^{(1)}$	31770	215000	$y^{(1)}$
$x^{(2)}$	11622	105000	$y^{(2)}$
$x^{(3)}$	14267	172000	$y^{(3)}$
$x^{(4)}$	11160	244000	$y^{(4)}$
$x^{(5)}$	13830	189900	$y^{(5)}$

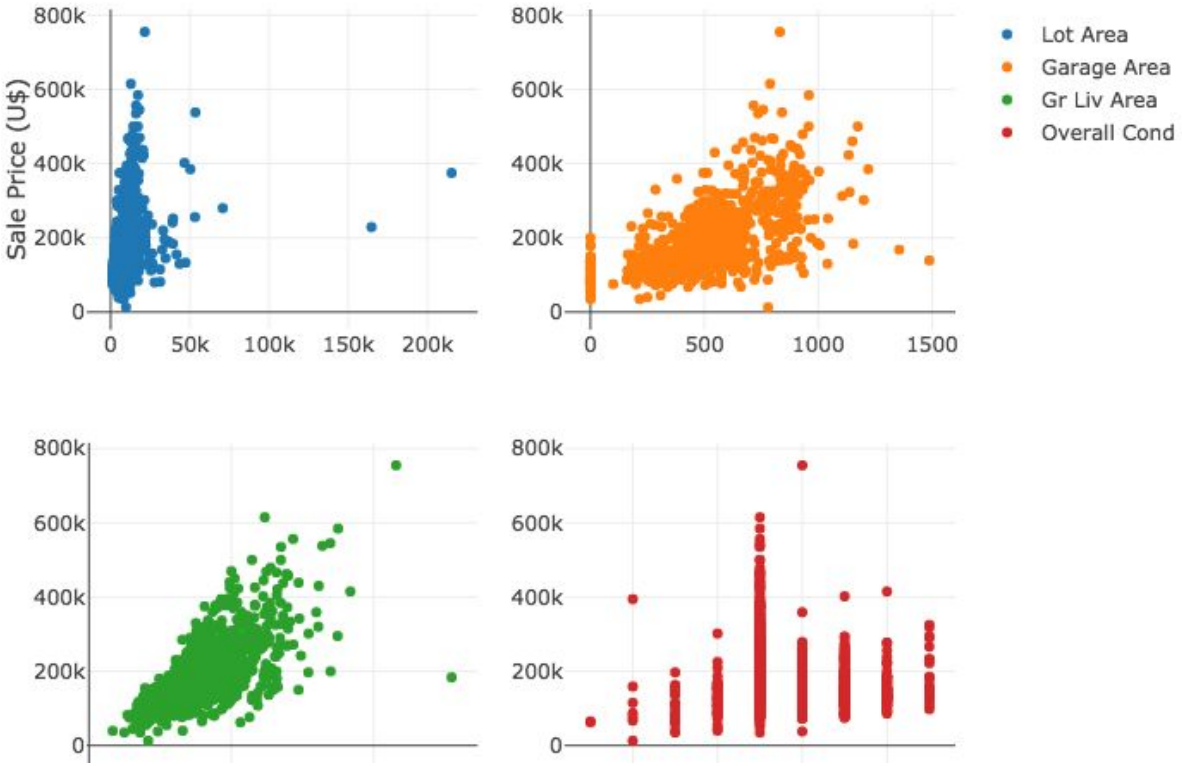
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)}\right]^2$$

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

Idea:

- choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples $(x^{(i)}, y^{(i)})$
- minimize (θ_0, θ_1)

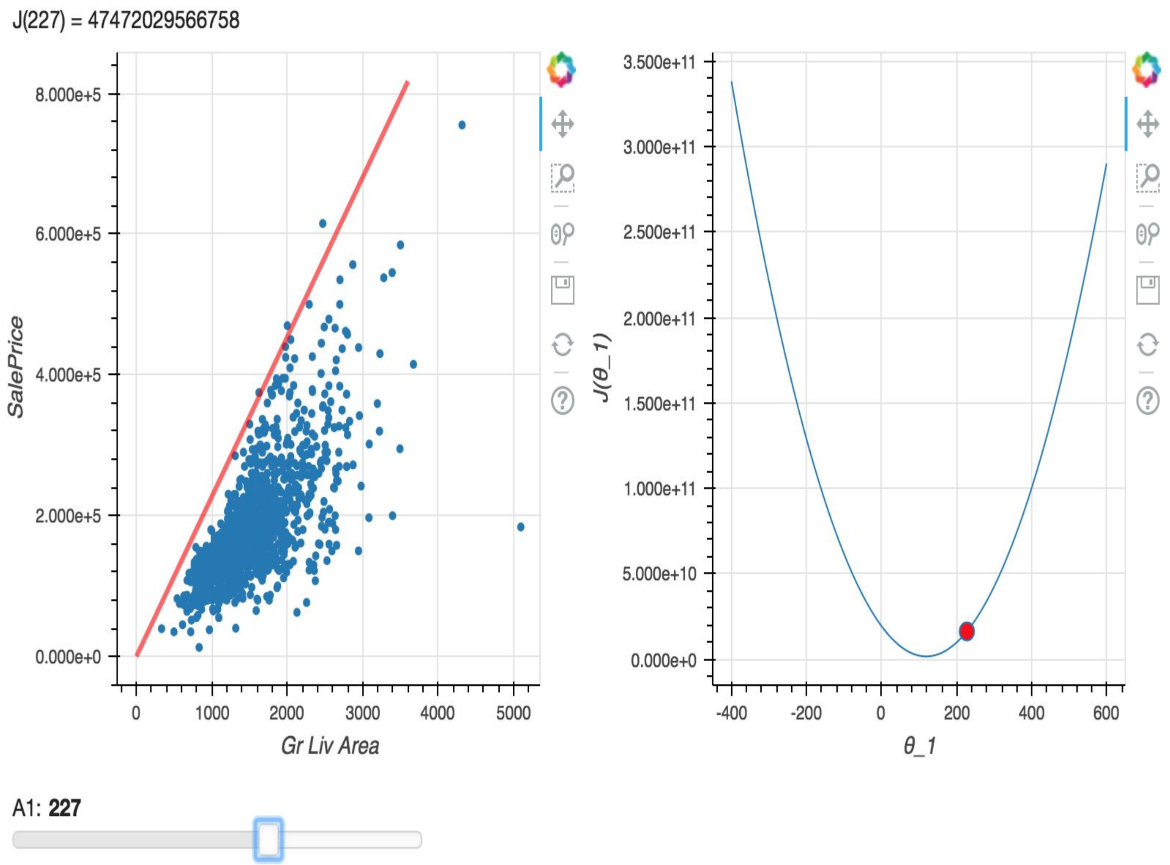
Cost Function[step #01] - Select the feature x



Cost Function[step #01] - Select the feature x_1

```
train[['Garage Area', 'Gr Liv Area', 'Overall Cond', 'Lot Area','SalePrice']].corr()
```

	Garage Area	Gr Liv Area	Overall Cond	Lot Area	SalePrice
Garage Area	1.000000	0.473506	-0.145705	0.213122	0.625335
Gr Liv Area	0.473506	1.000000	-0.134157	0.248676	0.706364
Overall Cond	-0.145705	-0.134157	1.000000	-0.042415	-0.108979
Lot Area	0.213122	0.248676	-0.042415	1.000000	0.267714
SalePrice	0.625335	0.706364	-0.108979	0.267714	1.000000



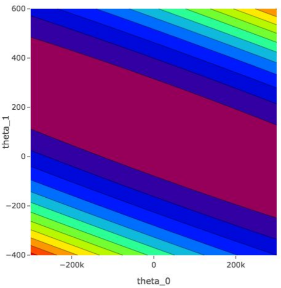
Cost Function Intuition #01 ($\theta_0 = 0$)

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

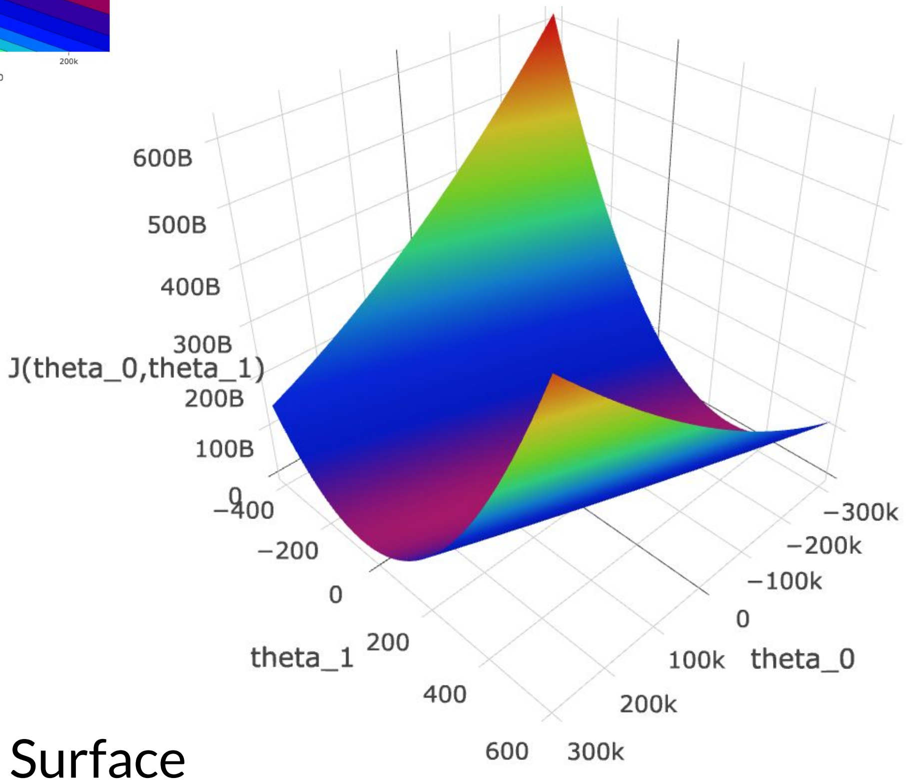
$$\hat{y} = h_\theta(x) = \theta_1 x$$

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

Cost Function Intuition #02 (linear reg. One var)



Contour



Surface

Cost Function

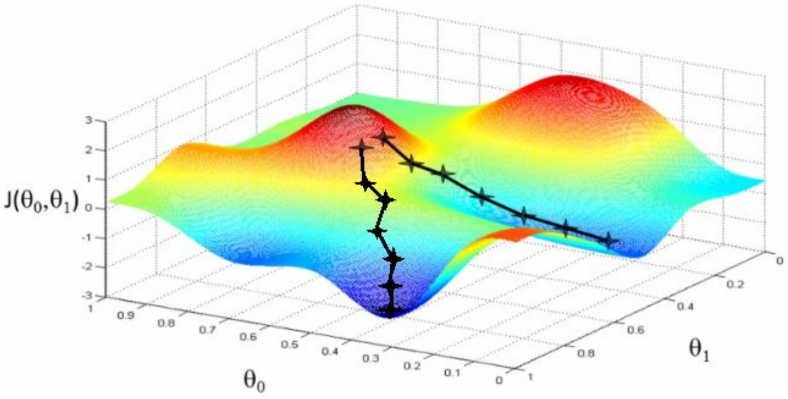
Intuition #02

(θ_0 and θ_1 are defined)

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

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

Gradient Descent (linear reg. One var)



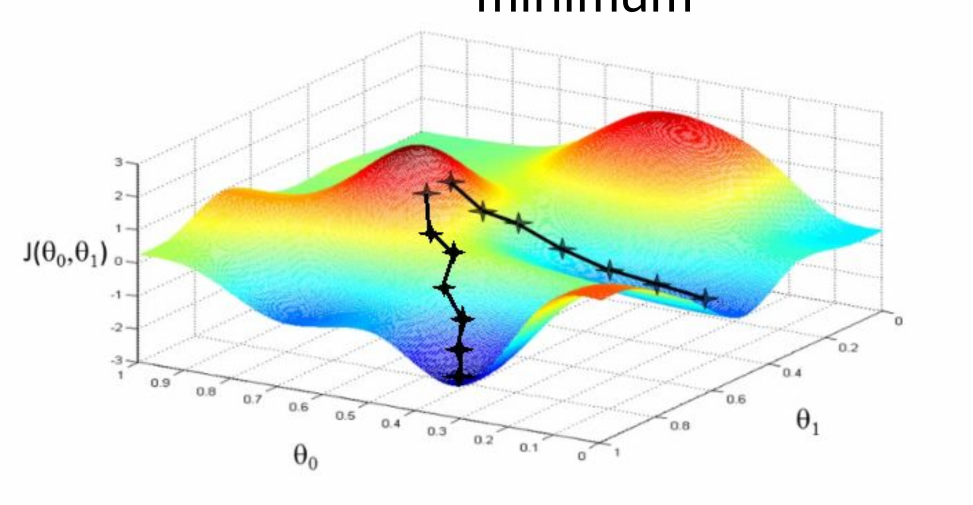
Algorithm - Idea

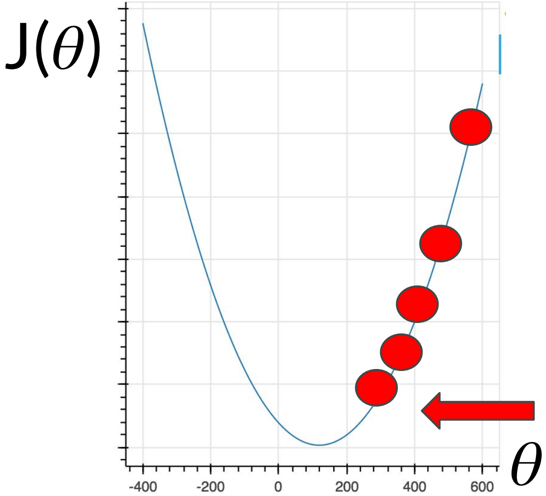
Have some function $J(\theta_0,\theta_1)$

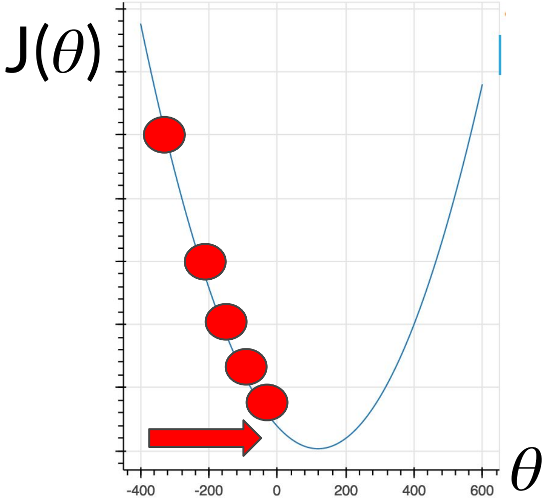
Want $\min_{\theta_0,\theta_1} J(\theta_0,\theta_1)$

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







repeat until converge {

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

}

α - learning rate

repeat until converge {

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

}

Correct update

$$aux_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$aux_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 = aux_0$$

$$\theta_1 = aux_1$$

Incorrect update

$$aux_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_0 = aux_0$$

$$aux_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_1 = aux_1$$

repeat until converge {

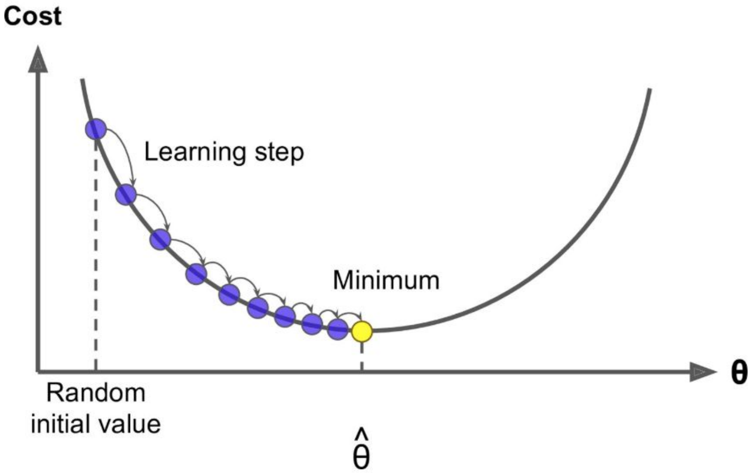
$$aux_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)}) \right]$$

$$aux_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)}) \right] x^{(i)}$$

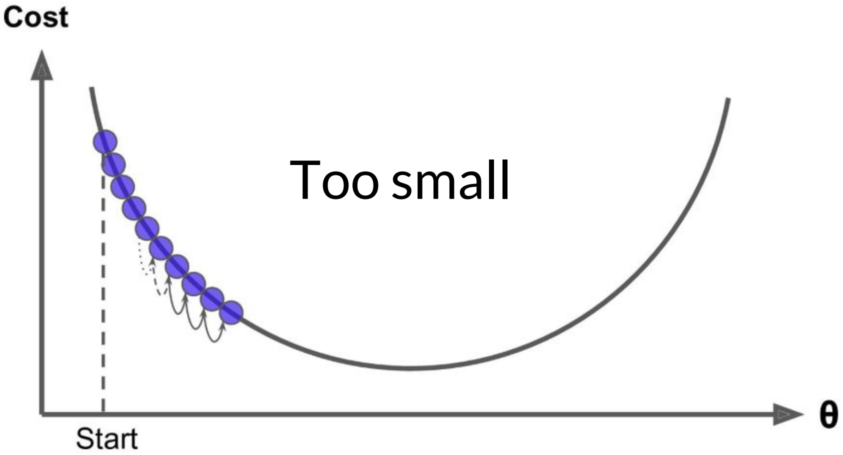
$$\theta_0 = aux_0$$

$$\theta_1 = aux_1$$

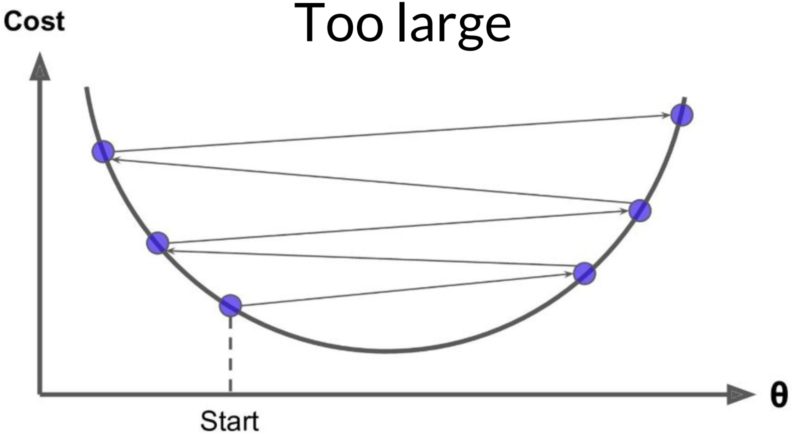
}



Too small



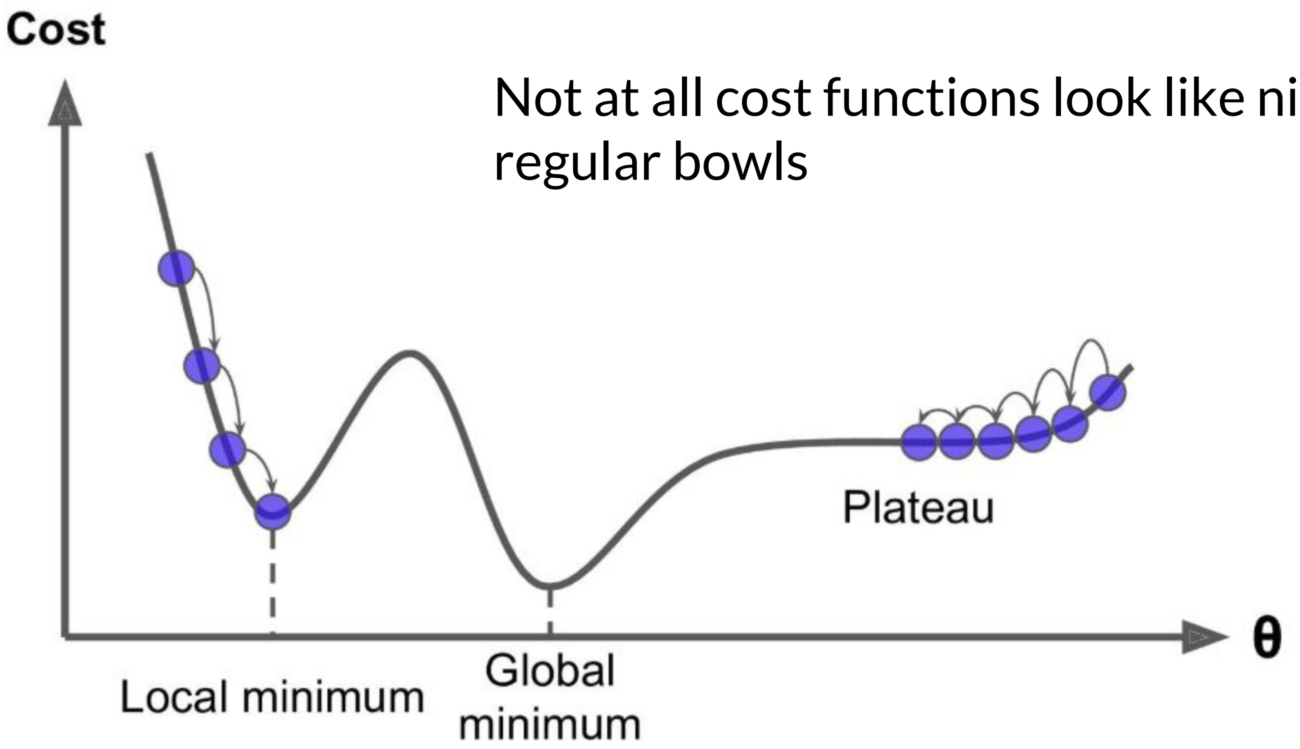
Too large



Learning rate tradeoff

Gradient Descent Pitfalls

Not at all cost functions look like nice regular bowls



cost function & gradient descent from linear algebra perspective

Hypothesis

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

Gr Liv Area	SalePrice
2480	205000
1829	237000
2673	249000
1005	133500
1768	224900

Export to plot.ly »

$$hypothesis = \begin{bmatrix} 1 & 2480 \\ 1 & 1829 \\ 1 & 2679 \\ 1 & 1005 \\ 1 & 1768 \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} 2480\,\theta_1 + \theta_0 \\ 1829\,\theta_1 + \theta_0 \\ 2679\,\theta_1 + \theta_0 \\ 1005\,\theta_1 + \theta_0 \\ 1768\,\theta_1 + \theta_0 \end{bmatrix}$$

```python
def cost_function(X, y, theta):
    return np.sum(np.square(np.matmul(X, theta) - y)) / (2 * len(y))
```

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

Gr Liv Area	SalePrice
2480	205000
1829	237000
2673	249000
1005	133500
1768	224900

Export to plot.ly »

$$J(\theta_0, \theta_1) = \frac{1}{2 \times 5} \sum \left(\begin{bmatrix} 2480\ \theta_1 + \theta_0 \\ 1829\ \theta_1 + \theta_0 \\ 2679\ \theta_1 + \theta_0 \\ 1005\ \theta_1 + \theta_0 \\ 1768\ \theta_1 + \theta_0 \end{bmatrix} - \begin{bmatrix} 205000 \\ 237000 \\ 249000 \\ 133500 \\ 224900 \end{bmatrix} \right)^2$$

```python
def gradient_descent(X, y, alpha, iterations, theta):
    m = len(y)
    all_thetas = [theta]

    for i in range(iterations):
        t0 = theta[0] - (alpha / m) * np.sum(np.dot(X, theta) - y)
        t1 = theta[1] - (alpha / m) * np.sum((np.dot(X, theta) - y) * X[:,1])
        theta = np.array([t0, t1])
        all_thetas.append([t0,t1])

    return theta, np.array(all_thetas)
```

- Involves calculations over the full training set X at each gradient step
- It uses the whole batch of training data at every step.
- This is why the algorithm called **Batch Gradient Descent**

repeat until converge {

$$aux_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right]$$

$$aux_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right] x^{(i)}$$

$$\theta_0 = aux_0$$

$$\theta_1 = aux_1$$

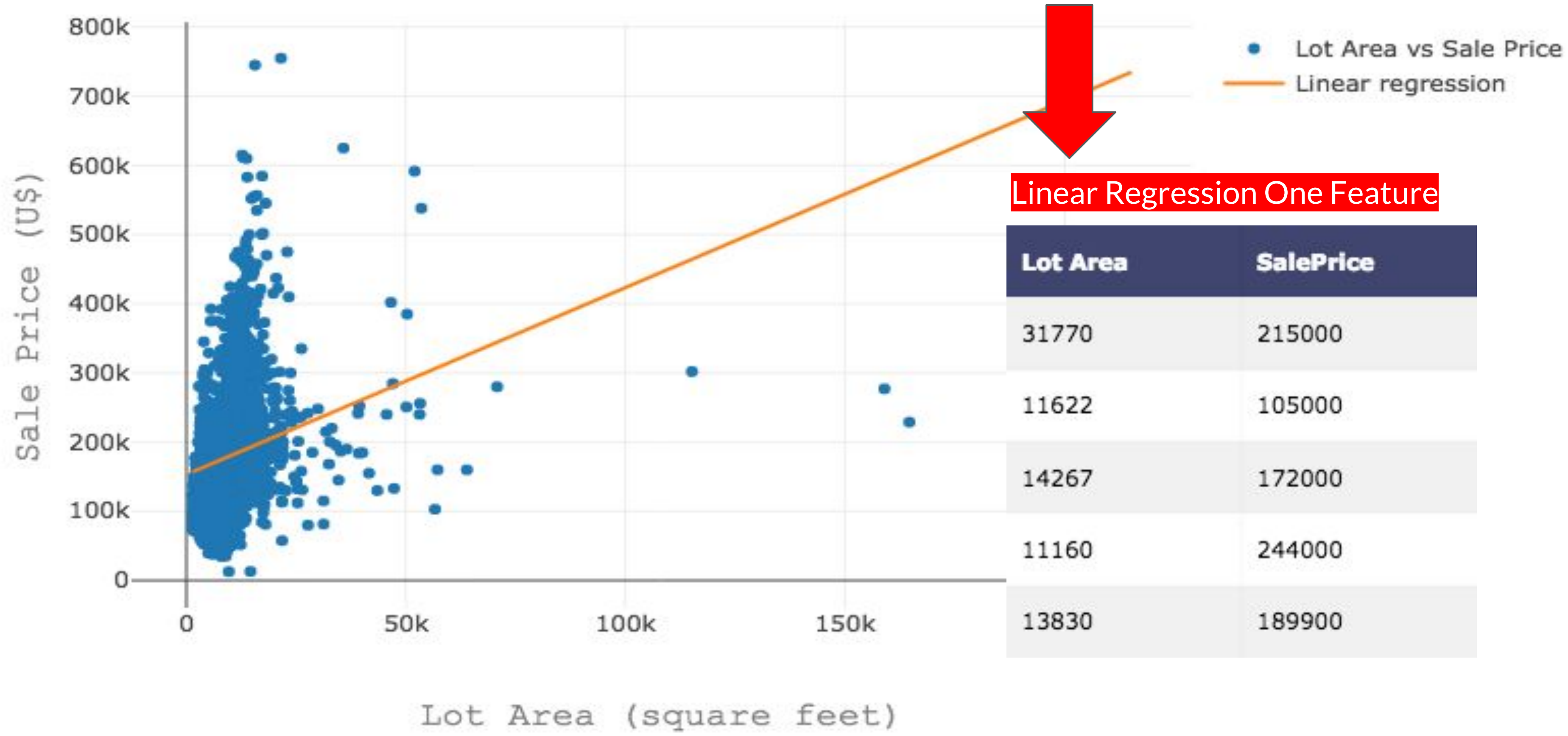
}

2_The linear Regression Model.ipynb

PREVIOUSLY ON...

lesson #18

Linear Regression - Housing Price



Linear Regression One Feature

Lot Area	SalePrice
31770	215000
11622	105000
14267	172000
11160	244000
13830	189900

Training Set (m instances)

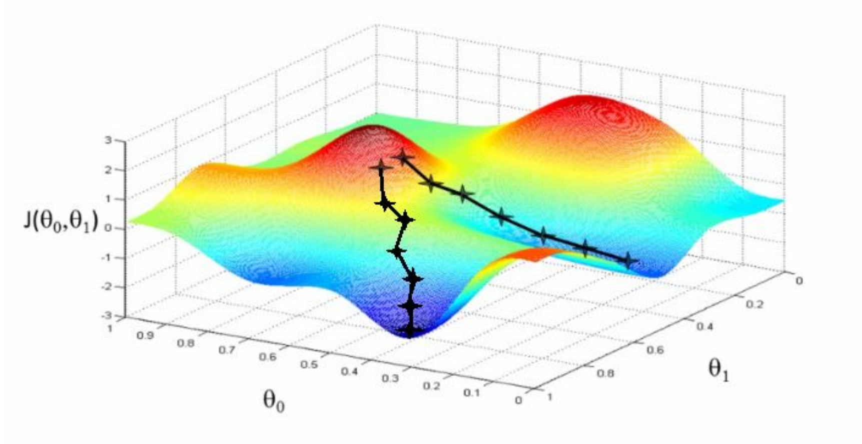
	Lot Area	SalePrice	
$x^{(1)}$	31770	215000	$y^{(1)}$
$x^{(2)}$	11622	105000	$y^{(2)}$
$x^{(3)}$	14267	172000	$y^{(3)}$
$x^{(4)}$	11160	244000	$y^{(4)}$
$x^{(5)}$	13830	189900	$y^{(5)}$

Cost Function

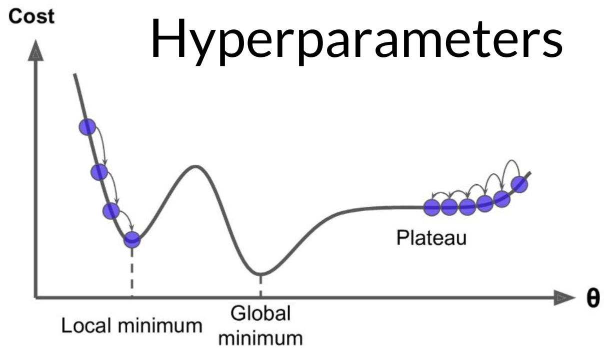
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right]^2$$

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

Gradient Descent



Hyperparameters



- We are going to start by covering linear regression
  - Multiple variables
- We discuss the application of linear regression to **housing price prediction**

Linear Regression with Multiple Variables

Notation:

- m - number of training examples
- n - number of features
- $x^{(i)}$ - input features of i^{th} training example
- $x_j^{(i)}$ - value of feature j in i^{th} training example
- $y^{(i)}$ - target value of i^{th} training examples

$$x^{(2)} = \begin{bmatrix} 11622 \\ 5 \\ 1961 \\ 2010 \\ 105000 \end{bmatrix}$$

$$x_3^{(2)} = 1961$$

n = 4

m = 5

x_1 Lot Area	x_2 Overall Qual	x_3 Year Built	x_4 Yr Sold	y SalePrice
31770	6	1960	2010	215000
11622	5	1961	2010	105000
14267	6	1958	2010	172000
11160	7	1968	2010	244000
13830	5	1997	2010	189900

Hypothesis (previously)

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

Multivariable case

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

For convenience of notation, define $x_0=1$. In other words:

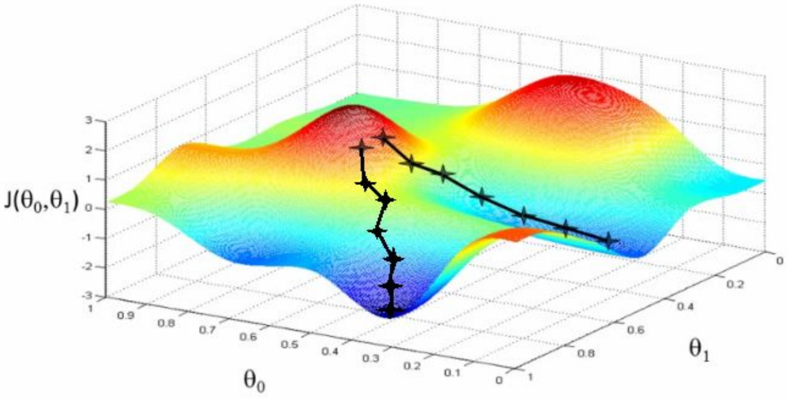
$x_0^{(i)}=1$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$h_\theta(x) = \begin{bmatrix} \theta_0 & \theta_1 & \theta_1 & \dots & \theta_n \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$

Gradient Descent
(linear reg. multiple variables)



Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \theta_2, \ldots, \theta_n$

Cost function:

$$J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Gradient descent:

repeat {

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

} Simultaneously update for every j (0 to n)

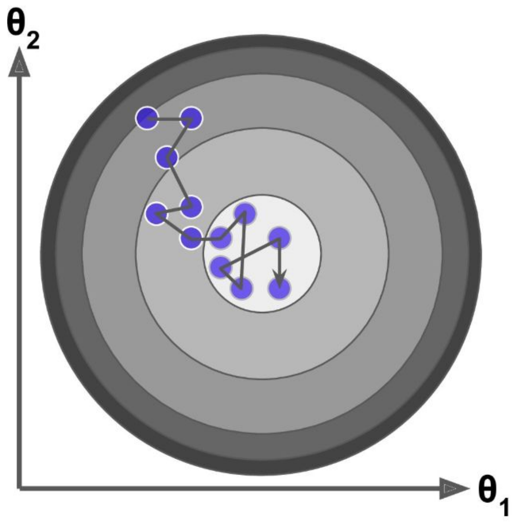
Gradient descent:

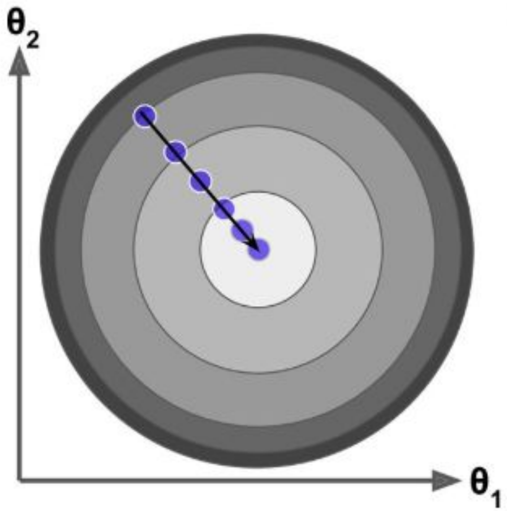
repeat until the convergence {

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad \text{for } j = 0, \ldots, n$$

}

Gradient Descent: trick #1 - Feature Scaling





Gradient Descent: trick #1 - Feature Scaling

Z-Score or Standardization

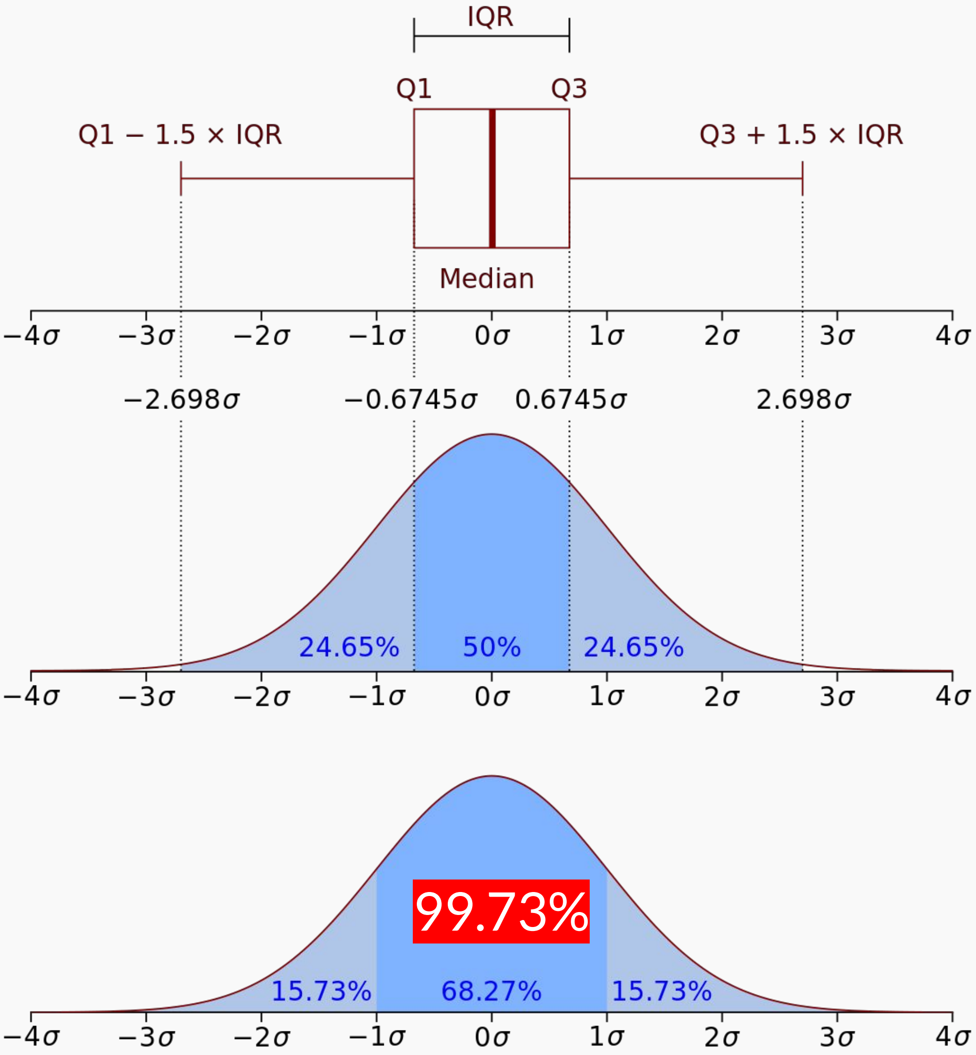
$$z = \frac{x - \mu}{\sigma}$$

μ - 0
σ - 1

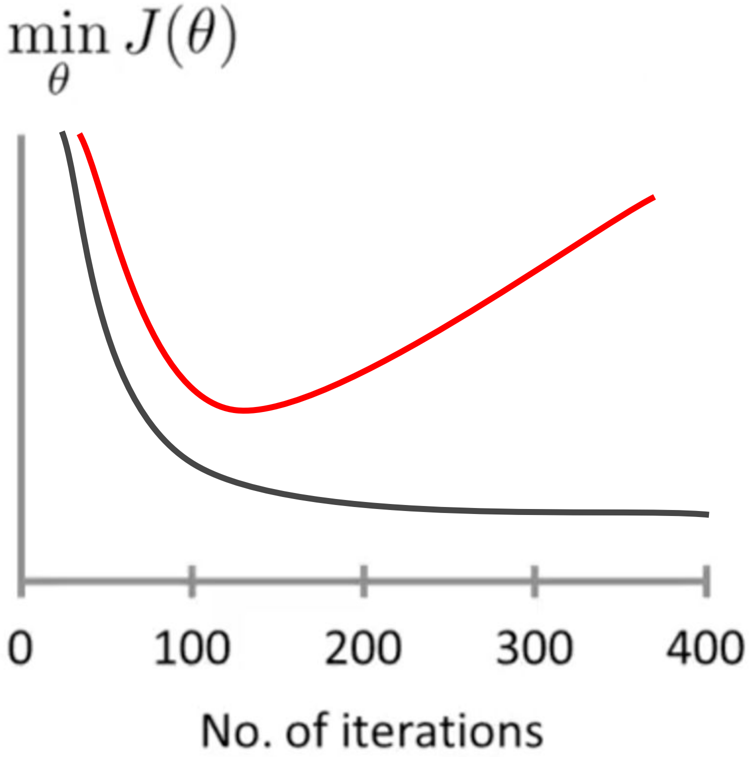
Min-Max Scaling

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

range (0,1)


IQR
Q1
Q3
Q1 − 1.5 × IQR
Q3 + 1.5 × IQR
Median
−4σ −3σ −2σ −1σ 0σ 1σ 2σ 3σ 4σ
−2.698σ
−0.6745σ
0.6745σ
2.698σ
24.65%
50%
24.65%
−4σ −3σ −2σ −1σ 0σ 1σ 2σ 3σ 4σ
99.73%
15.73%
68.27%
15.73%
−4σ −3σ −2σ −1σ 0σ 1σ 2σ 3σ 4σ

Gradient Descent: trick #2 - Debugging α



1. Make a plot with number of iterations on the x-axis.
2. Now plot the cost function, J(θ) over the number of iterations of gradient descent.
3. If J(θ) ever increases, then you probably need to decrease α.
4. It has been proven that if learning rate α is sufficiently small, then J(θ) will decrease on every iteration.
5. Automatic convergence test

Gradient Descent: trick #2 - Debugging α

- If α is too small:
  - Slow convergence
- If α is too large:
  - J(*θ*) may not decrease on every iteration;
  - J(*θ*) may not converge.

To choice α:
…, 0.0001, ….., 0.001, …, 0.01, …, 0.1, …, 1, …, 10, ...

normal equation: method to solve for θ analytically

Normal Equation

Lot Area	Overall Qual	Year Built	Yr Sold	SalePrice
31770	6	1960	2010	215000
11622	5	1961	2010	105000
14267	6	1958	2010	172000
11160	7	1968	2010	244000
13830	5	1997	2010	189900

$$X\theta = y$$
$$X^T X\theta = X^T y$$
$$\theta = (X^T X)^{-1} X^T y$$

$$\begin{bmatrix} 1 & 31770 & 6 & 1960 & 2010 \\ 1 & 11622 & 5 & 1961 & 2010 \\ 1 & 14267 & 6 & 1958 & 2010 \\ 1 & 11160 & 7 & 1968 & 2010 \\ 1 & 13830 & 5 & 1997 & 2010 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix} = \begin{bmatrix} \theta_0 + 31770\theta_1 + 6\theta_2 + 1960\theta_3 + 2010\theta_4 \\ \theta_0 + 11622\theta_1 + 5\theta_2 + 1961\theta_3 + 2010\theta_4 \\ \theta_0 + 14267\theta_1 + 6\theta_2 + 1958\theta_3 + 2010\theta_4 \\ \theta_0 + 11160\theta_1 + 7\theta_2 + 1968\theta_3 + 2010\theta_4 \\ \theta_0 + 13830\theta_1 + 5\theta_2 + 1997\theta_3 + 2010\theta_4 \end{bmatrix}$$

There is **no need** to do feature scaling with the normal equation.

The following is a comparison of gradient descent and the normal equation:

Gradient Descent	**Normal Equation**
Need to choose alpha	No need to choose alpha
Needs many iterations	No need to iterate
O (kn^2)	O (n^3), need to calculate inverse of X^TX
Works well when n is large	Slow if n is very large

With the normal equation, computing the inversion has complexity $\mathcal{O}(n^3)$. So if we have a very large number of features, the normal equation will be slow. In practice, when n exceeds 10,000 it might be a good time to go from a normal solution to an iterative process.

If $X^T X$ is **noninvertible,** the common causes might be having :

- Redundant features, where two features are very closely related (i.e. they are linearly dependent)
- Too many features (e.g. m ≤ n). In this case, delete some features or use "regularization" (to be explained in a later lesson).

3. The linear Regression Model II.ipynb

- Section 2 - The Linear Regression Model
  - Hands on based Scikit-Learn
- Section 3 - Feature selection
  - Missing values, correlation, train and set models, removing low variance features
- Section 4 - Gradient Descent
- Section 5 - Ordinary Least Square
- Section 6 - Processing and Transform Features
- Sections 7 & 8 - Guided Project