EEC1509 - Machine Learning

Lesson #4 - Linear Regression One Variable

Ivanovitch Silva

August, 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**.

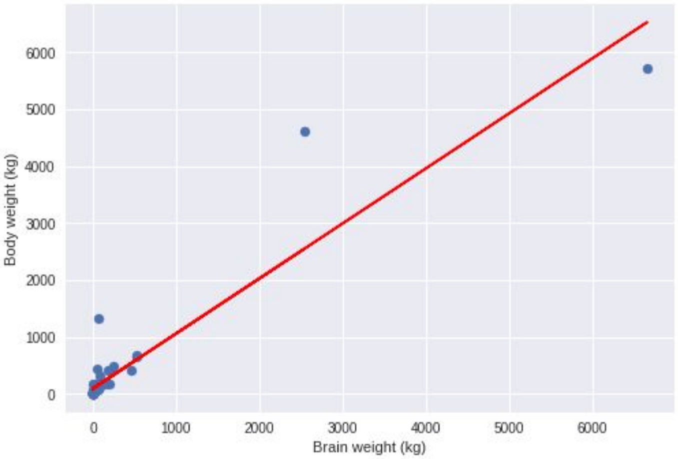
Regression

5 kilograms

200 kilograms

1.5 kilograms

????

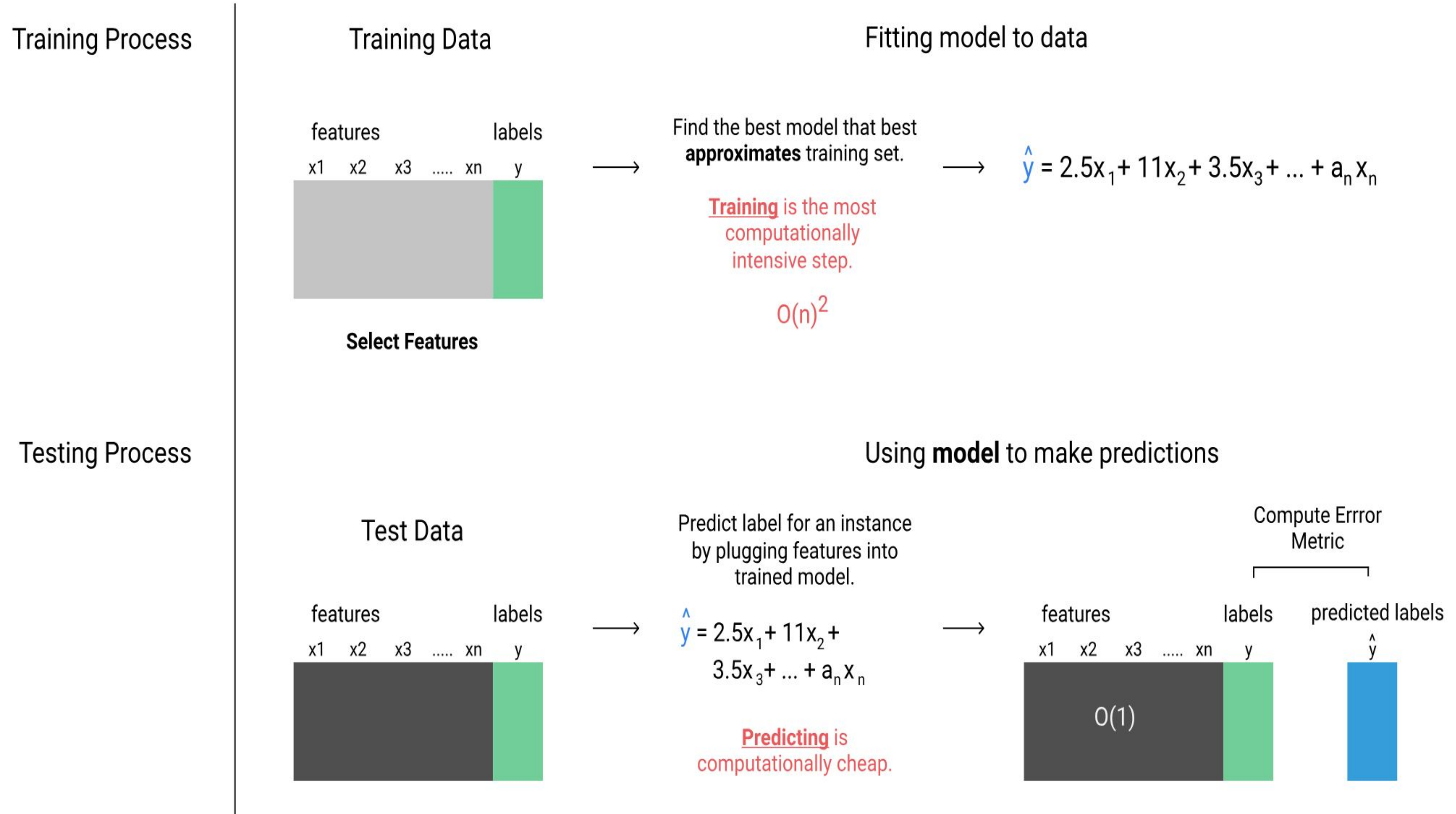



```python
import pandas as pd
from sklearn import linear_model
import matplotlib.pyplot as plt

#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 (15min)


Training Process
Training Data
Fitting model to data
features
labels
x1 x2 x3 ..... xn y
Find the best model that best **approximates** training set.
$\hat{y} = 2.5x_1 + 11x_2 + 3.5x_3 + \ldots + a_n x_n$
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
$\hat{y} = 2.5x_1 + 11x_2 + 3.5x_3 + \ldots + a_n x_n$
features
labels
predicted labels
x1 x2 x3 ..... xn y $\hat{y}$
O(1)
Predicting is computationally cheap.

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 (15min)
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

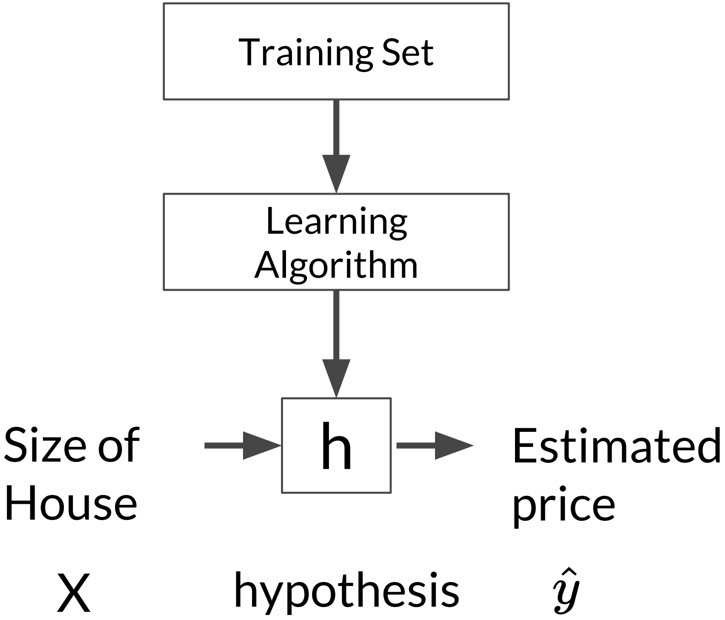
X	y
Lot Area	SalePrice
31770	215000
11622	105000
14267	172000
11160	244000
13830	189900

$m = 1465$

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

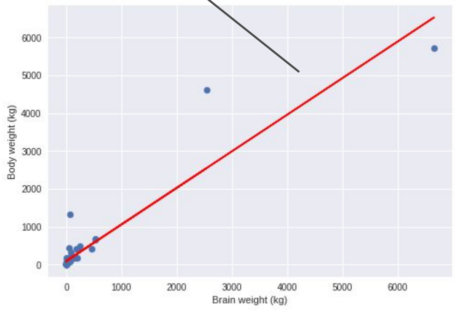
$(X^{(i)}, y^{(i)})$ = i^{th} training example

Model Representation (linear reg. one variable)

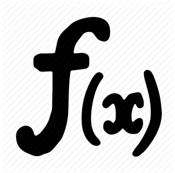


How do we represent h?

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



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 ?

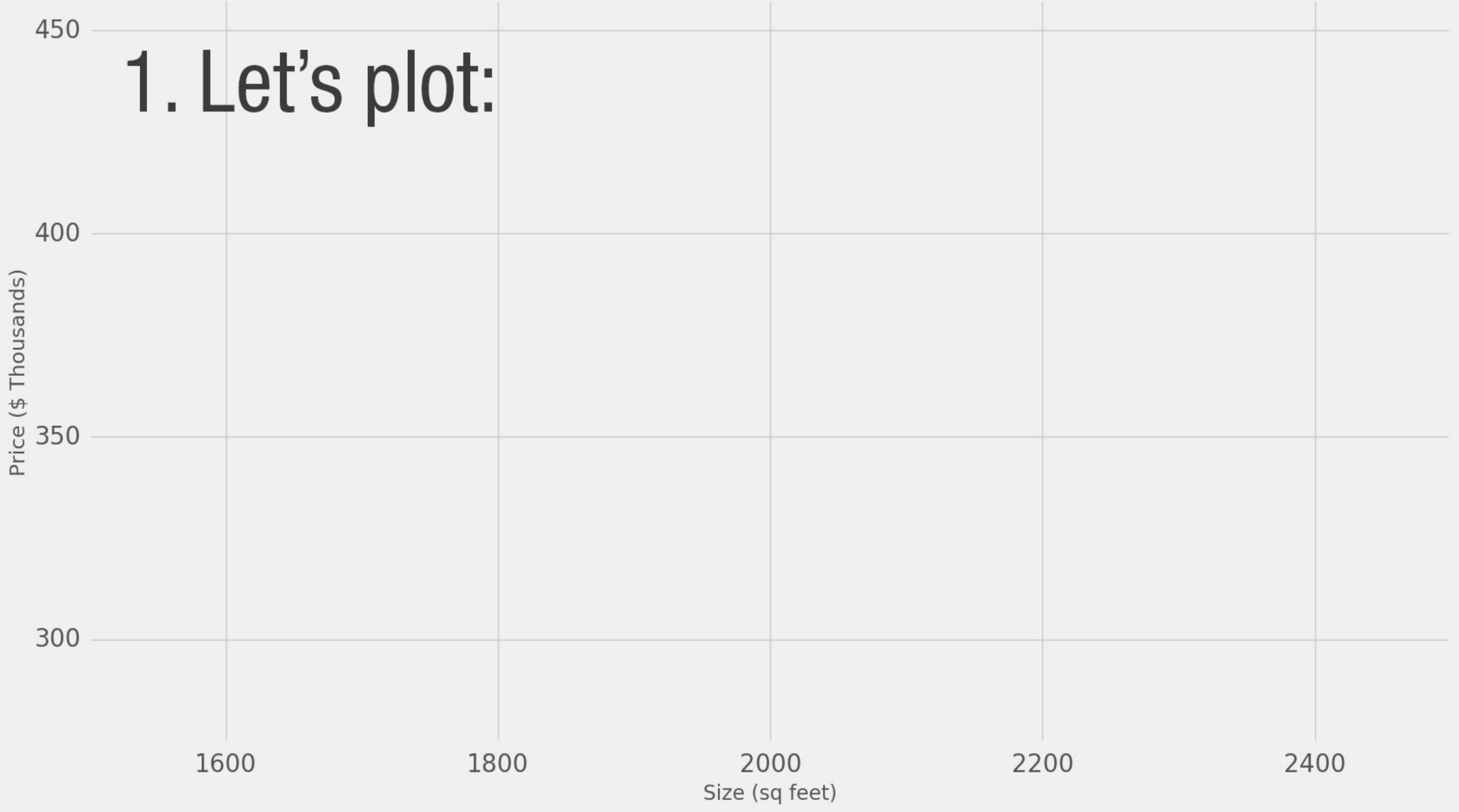
Training Set

$m = 1465$

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

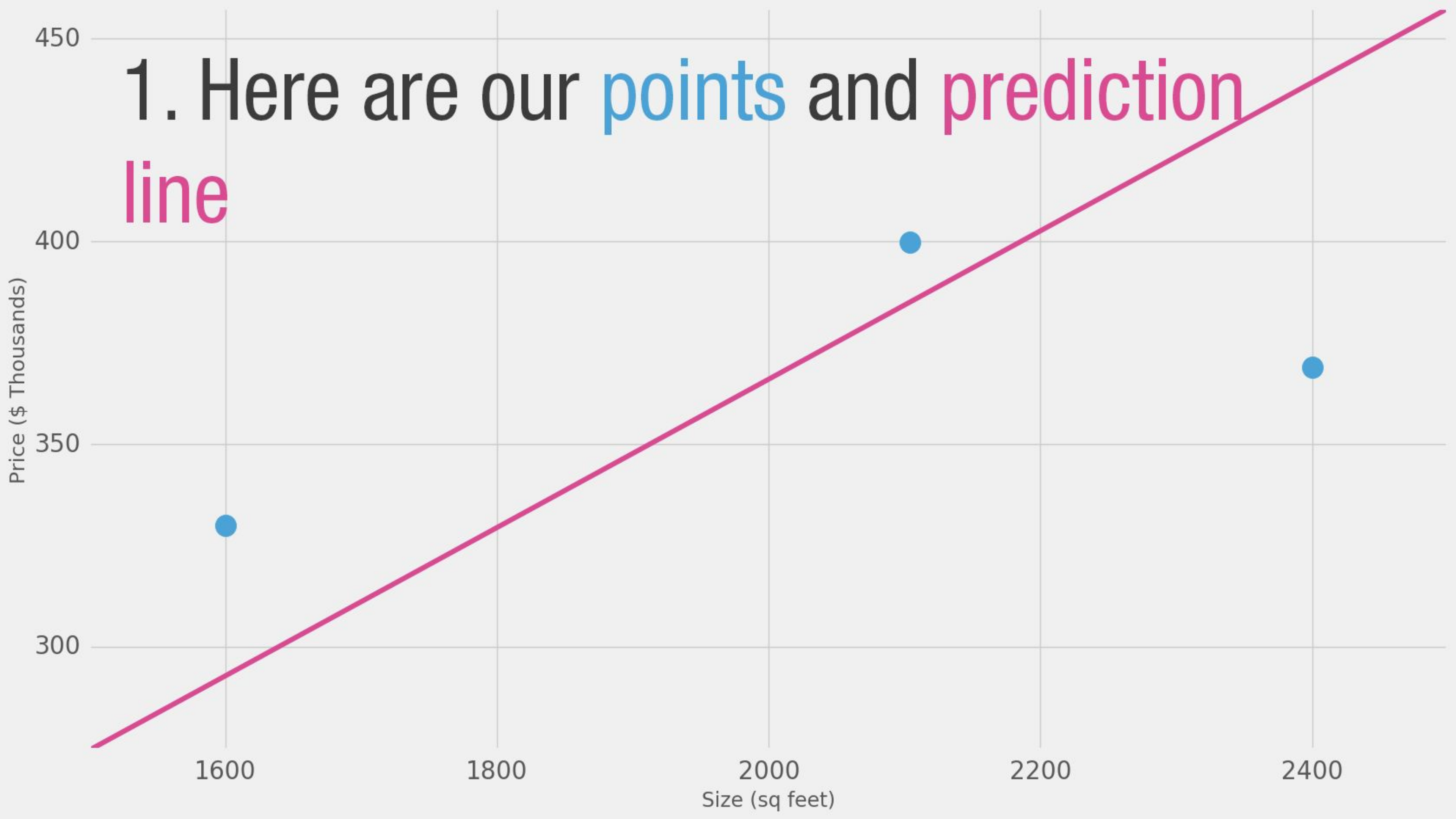
Cost Function Intuition #01 (linear reg. One var)

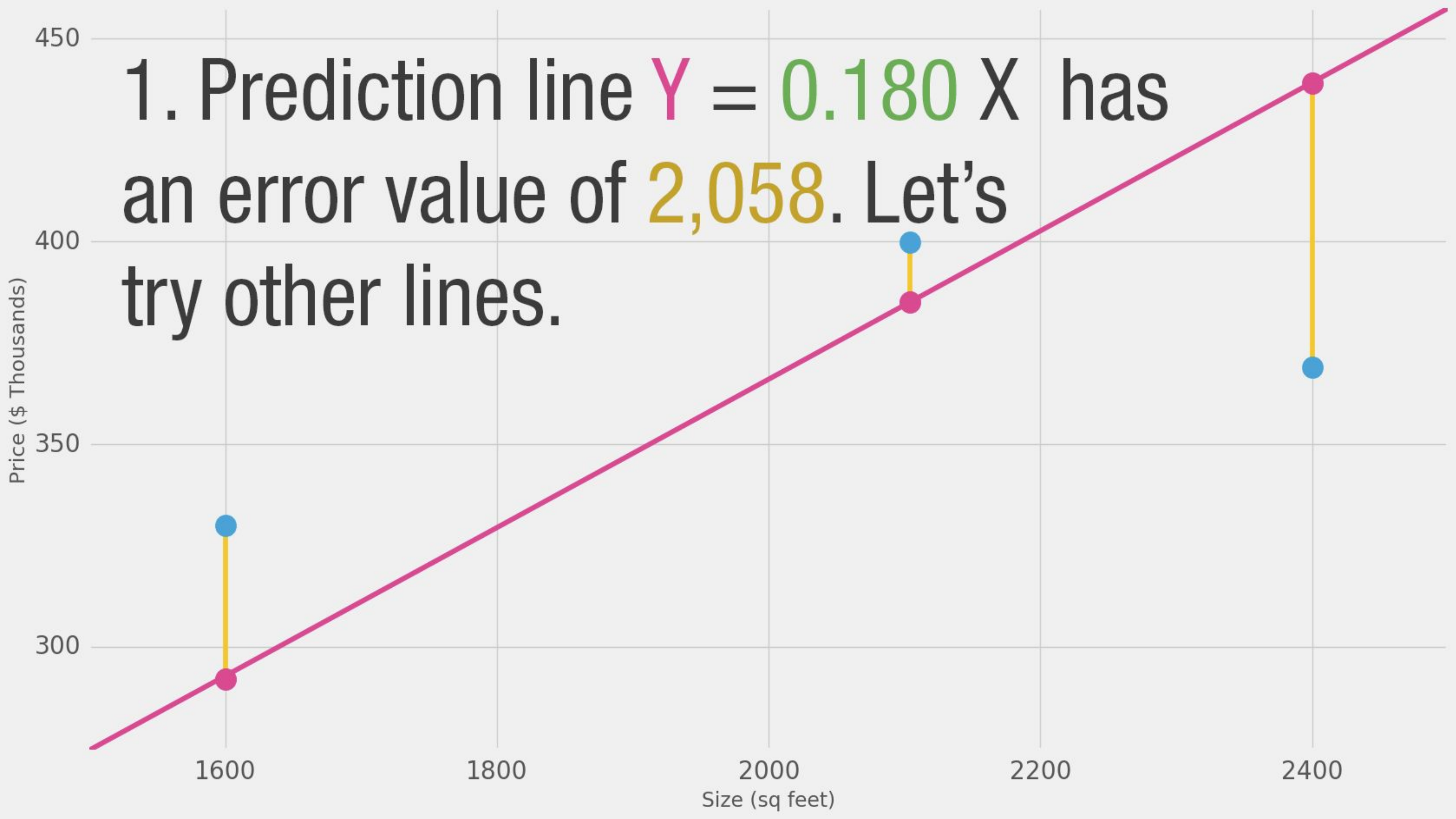
1. Let’s plot:



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

1. Here are our points and prediction line




1. Prediction line Y = 0.180 X has an error value of 2,058. Let's try other lines.
450
400
350
300
Price ($ Thousands)
1600
1800
2000
2200
2400
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	$y^{(5)}$

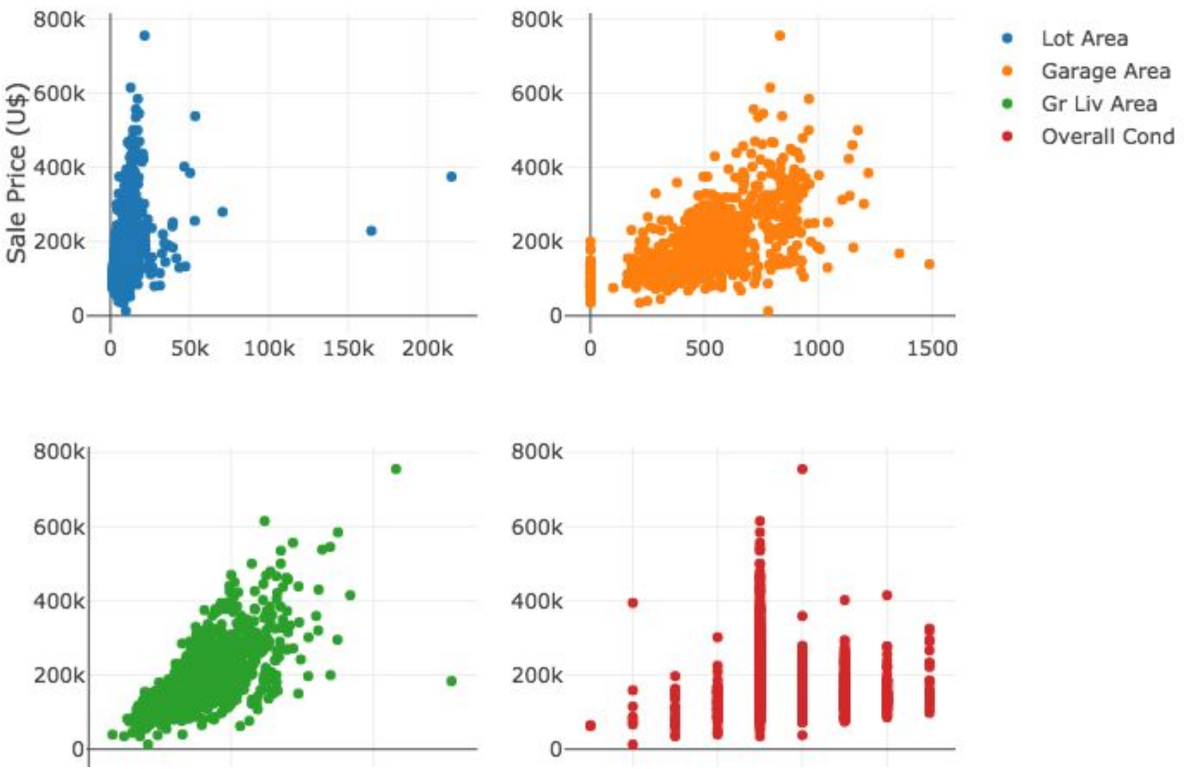
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)}\right]^2$$

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

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)

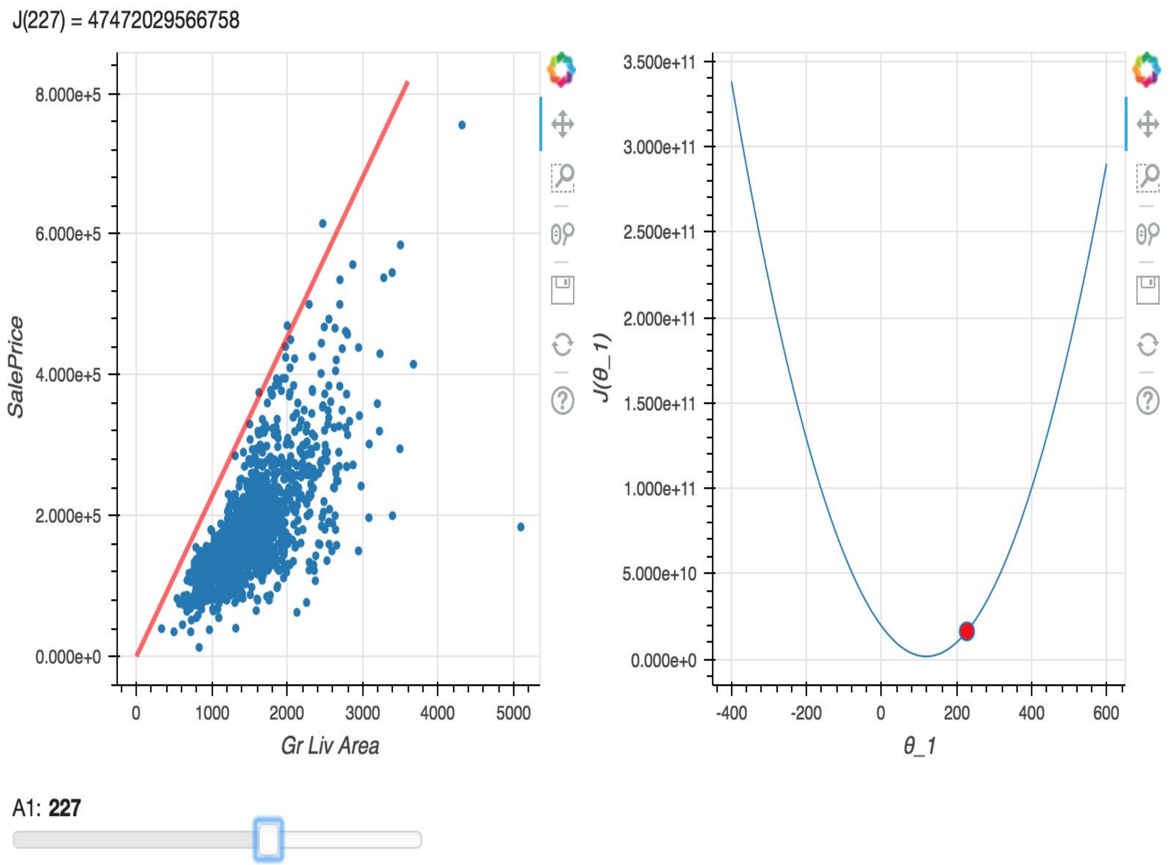
Cost Function[step #01] - Select the feature x



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

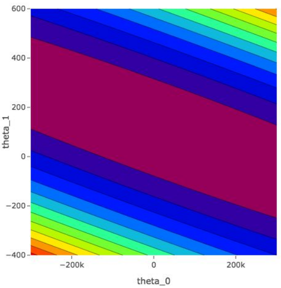
Intuition #01

$(\theta_0 = 0)$

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

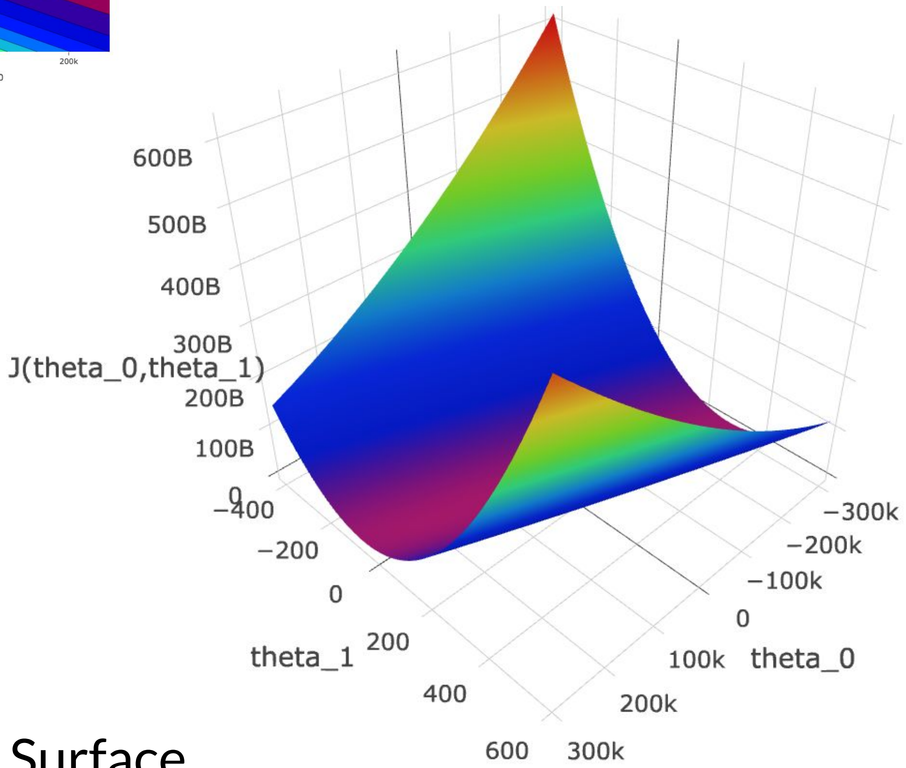
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left[\theta_1 x^{(i)} - y^{(i)}\right]^2$$

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



Contour

Cost Function
Intuition #02
(θ_0 and θ_1 are defined)

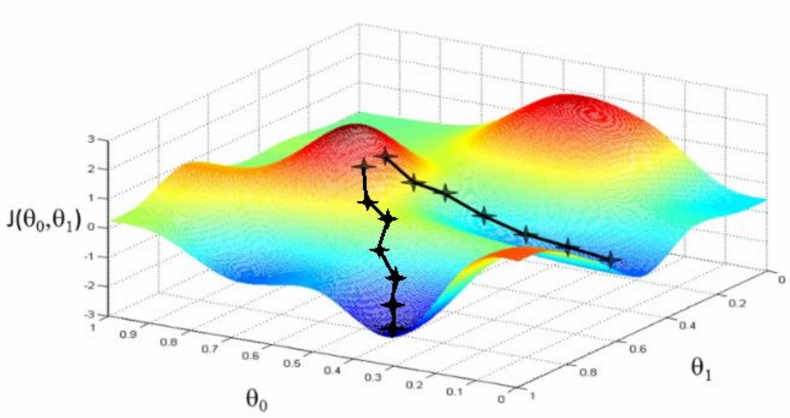


$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x$$

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

Surface

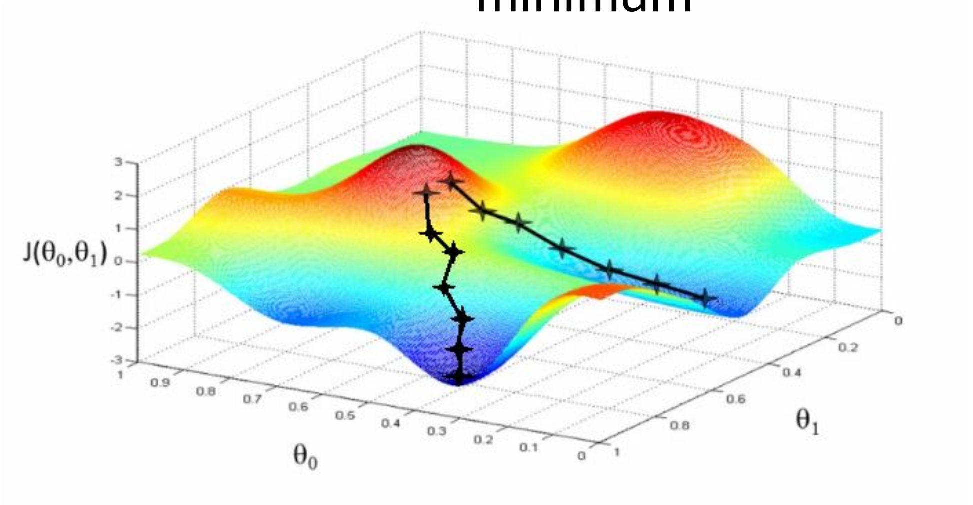
Gradient Descent (linear reg. One var)

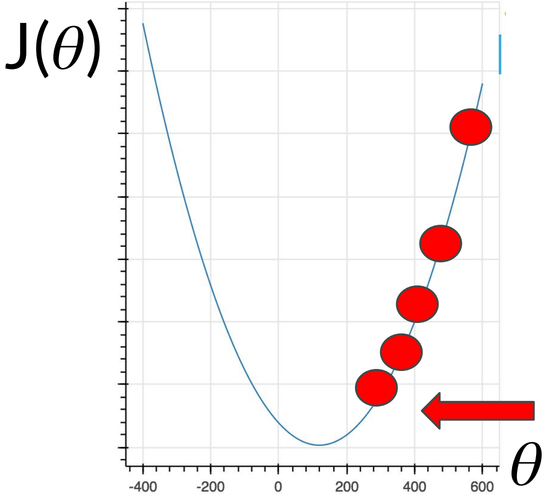


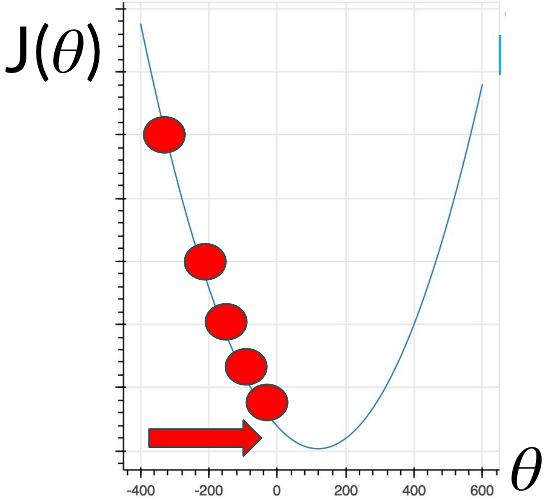
Algorithm - Idea

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 {

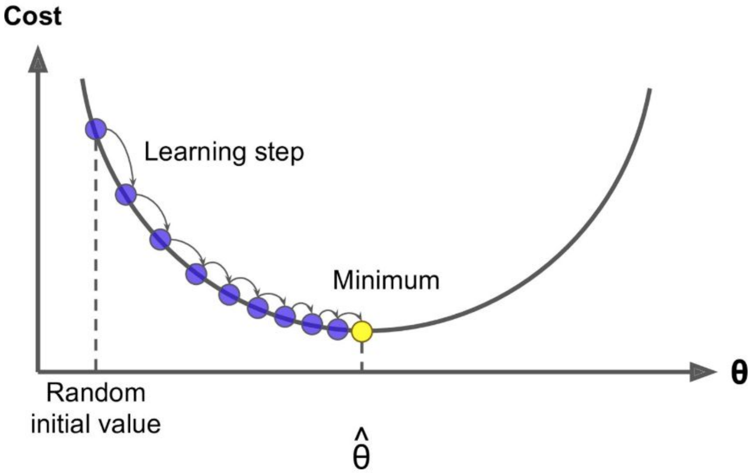
$$aux_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right]$$

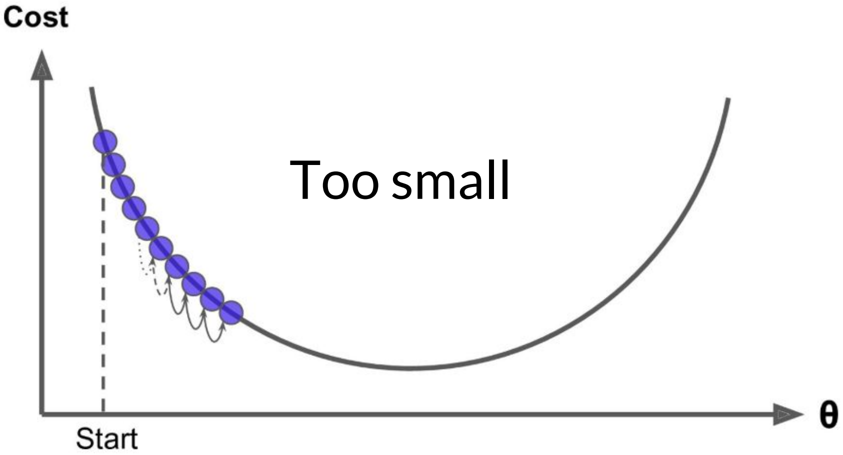
$$aux_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) - y^{(i)} \right] x^{(i)}$$

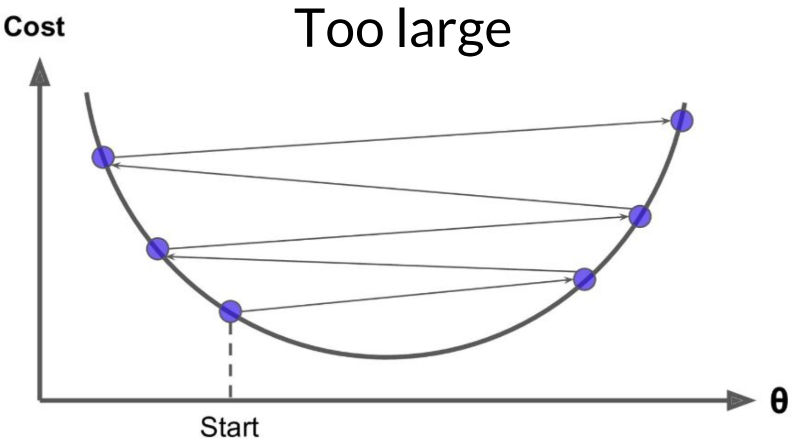
$\theta_0 = aux_0$

$\theta_1 = aux_1$

}



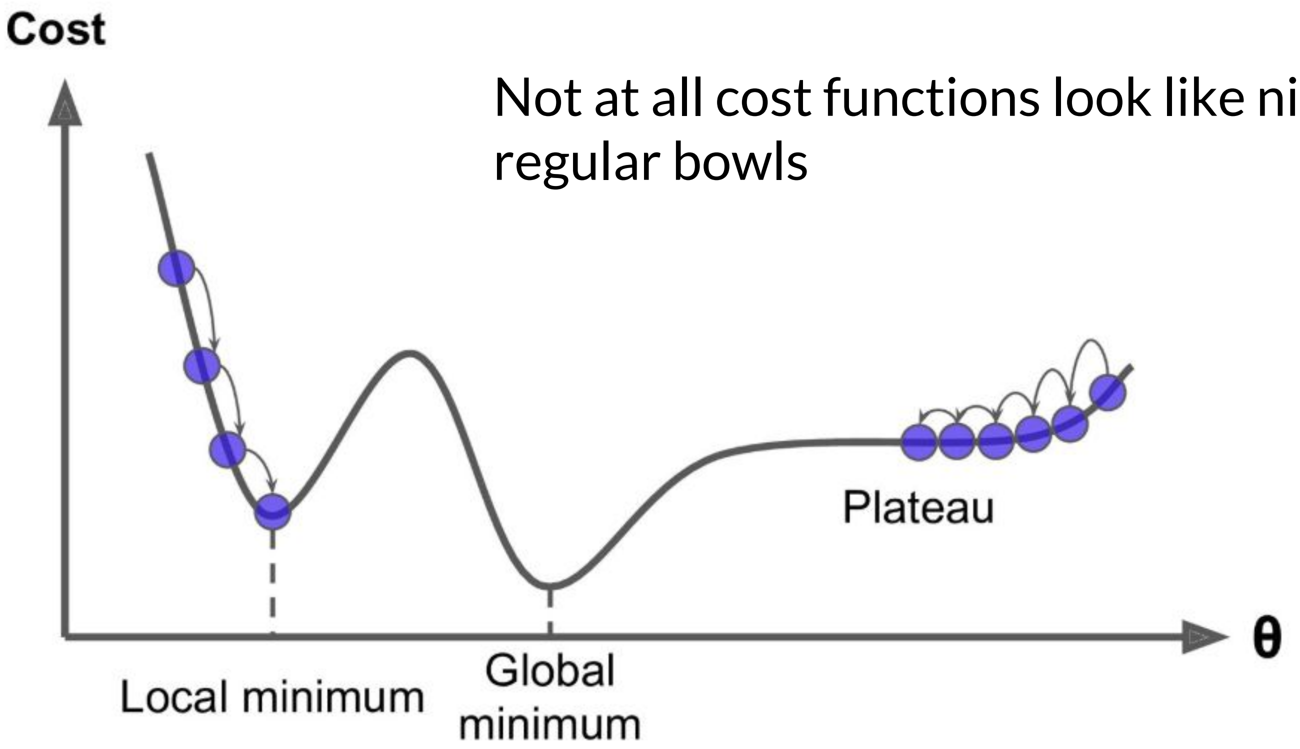




Learning rate tradeoff

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 {

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

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

$$\theta_0 = aux_0$$

$$\theta_1 = aux_1$$

}