Module 3, Day 2

Alireza Samar

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

MLDS
MACHINE LEARNING
FOR DATA SCIENCE

Goal

- Perform **linear regression** in its most basic form through simplest example
  - Predict house price with just single feature, i.e., house size

MLDS

Machine Learning Purpose

- **Predict** something given a set of **features** that influences prediction:
  - Predict house price given its size, location, configuration (features)

MLDS

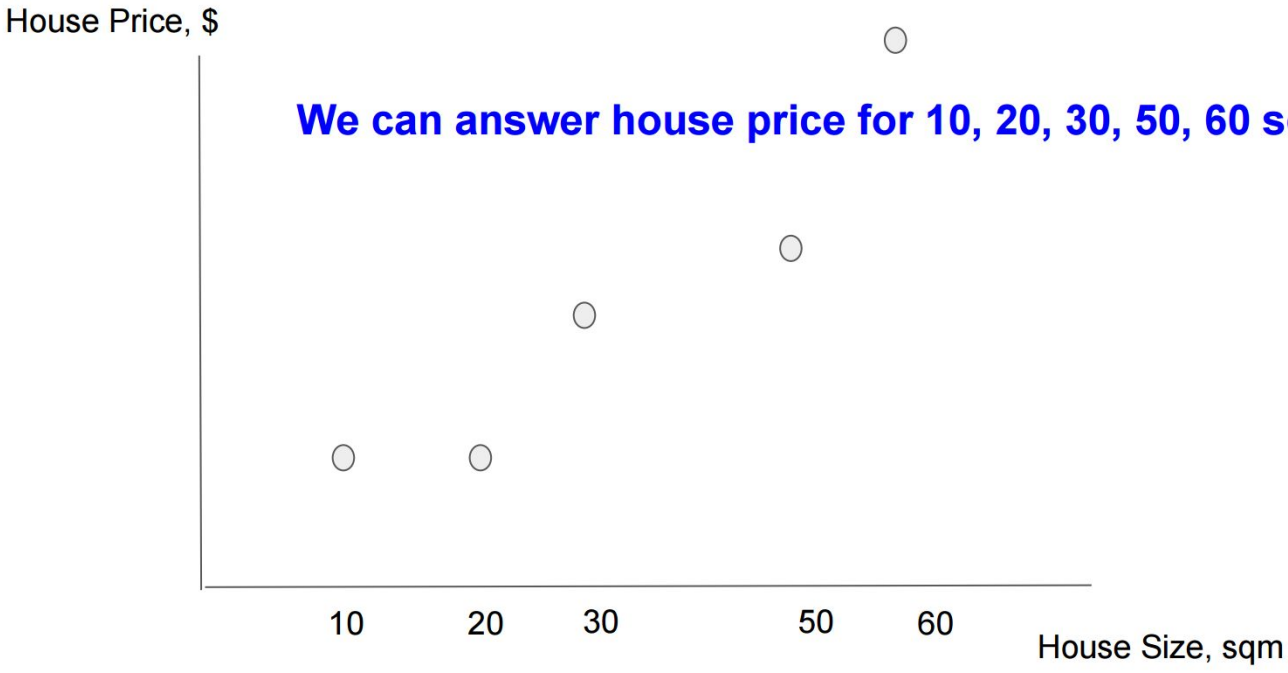
Machine Learning Training

Before predicting, we have to:

- Choose/Create a **model**
- **Train** the model to learn prediction with **data**

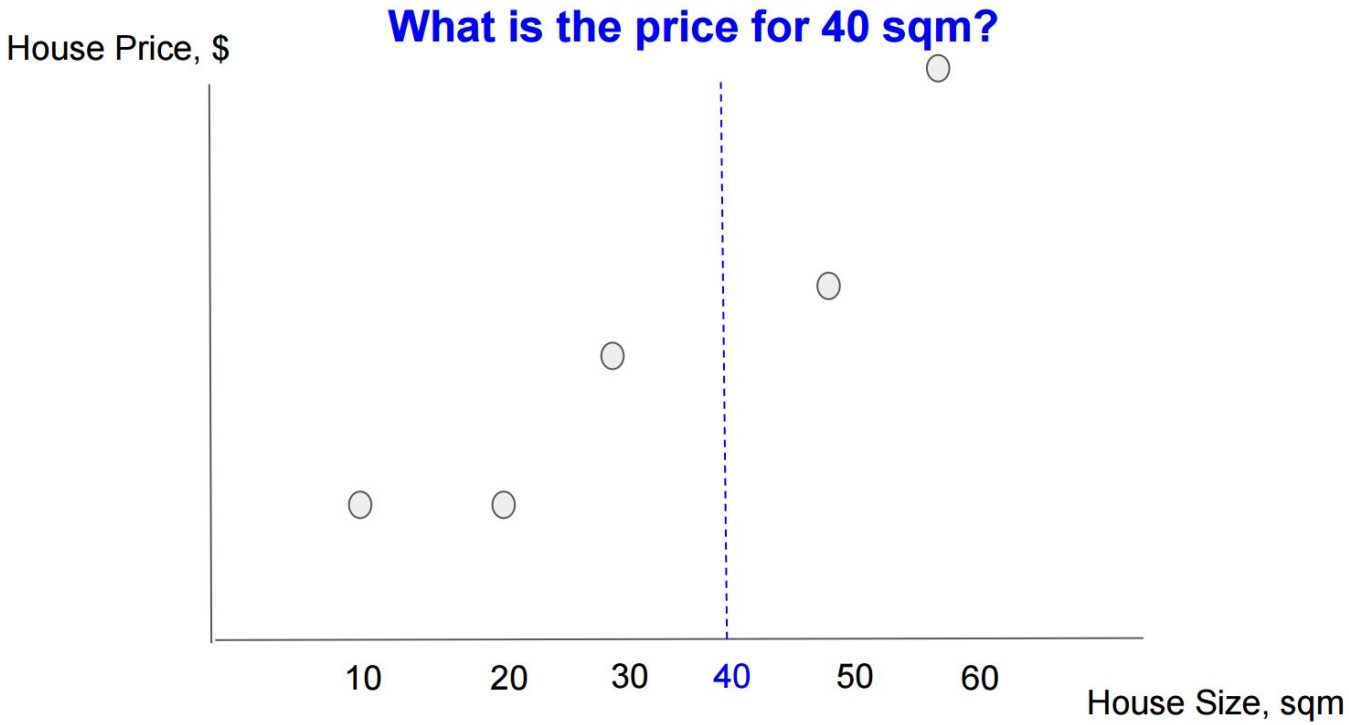
MLDS

What is the House Price?



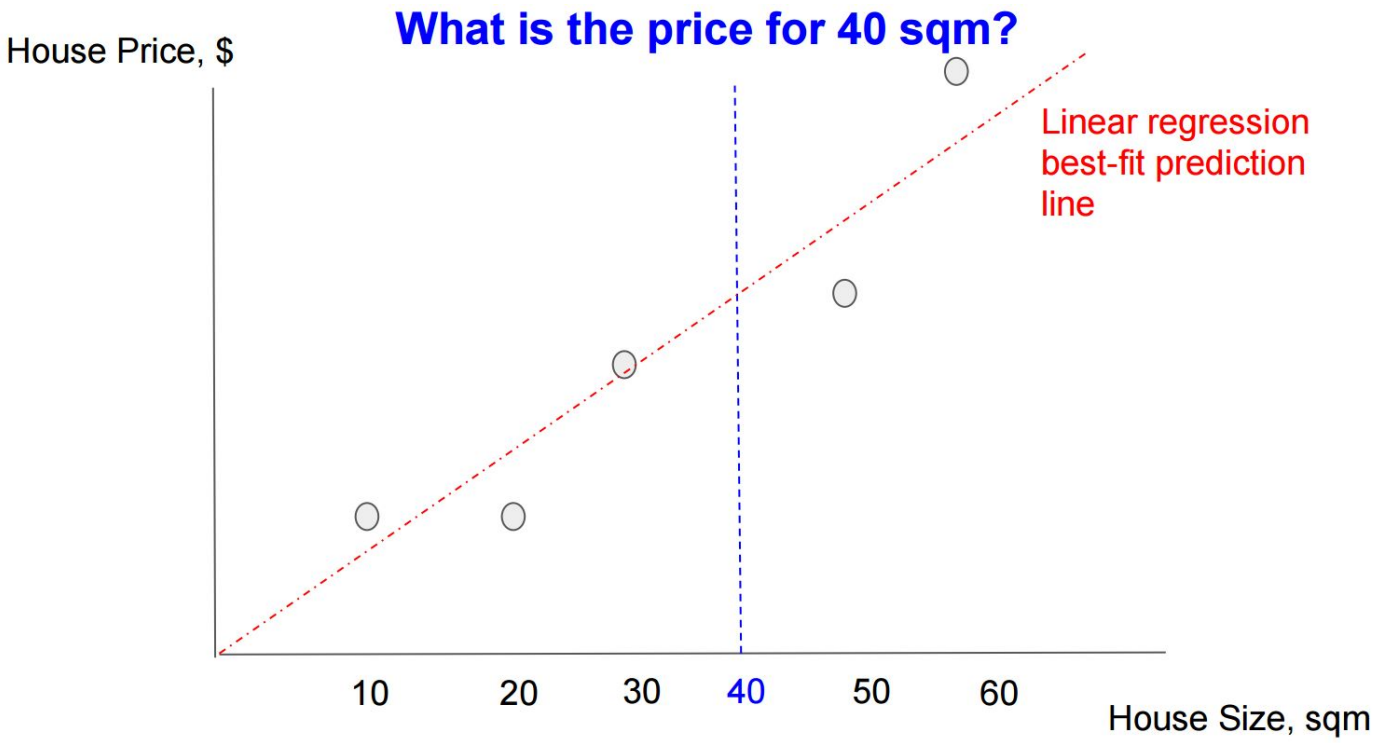
MLDS

What is the House Price?



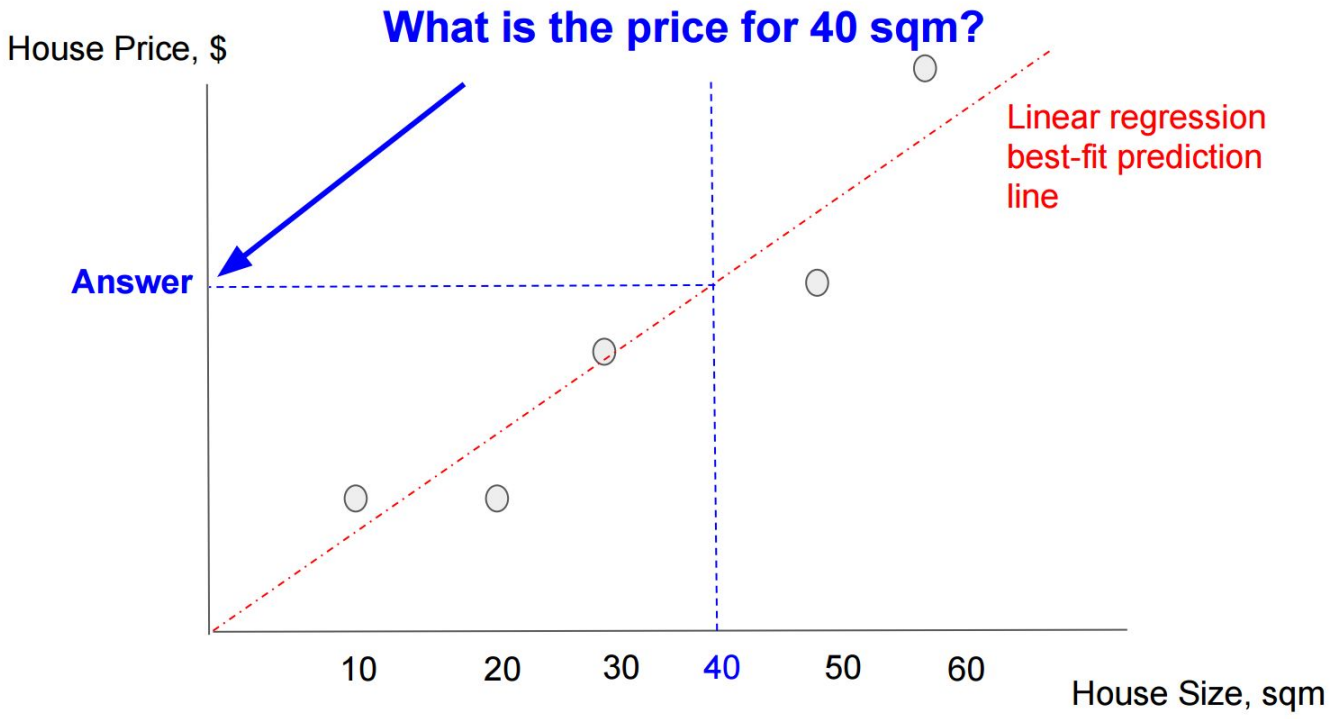
MLDS

What is the House Price?



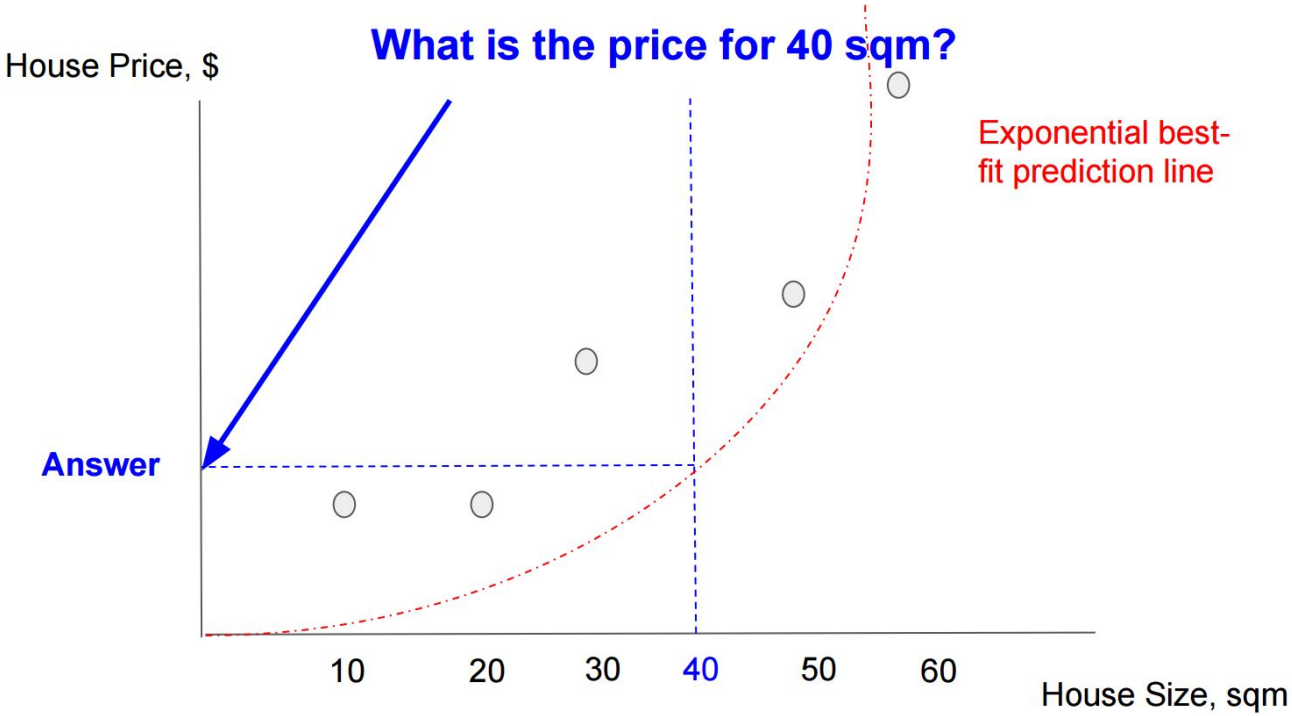
MLDS

What is the House Price?



MLDS

What is the House Price?



MLDS

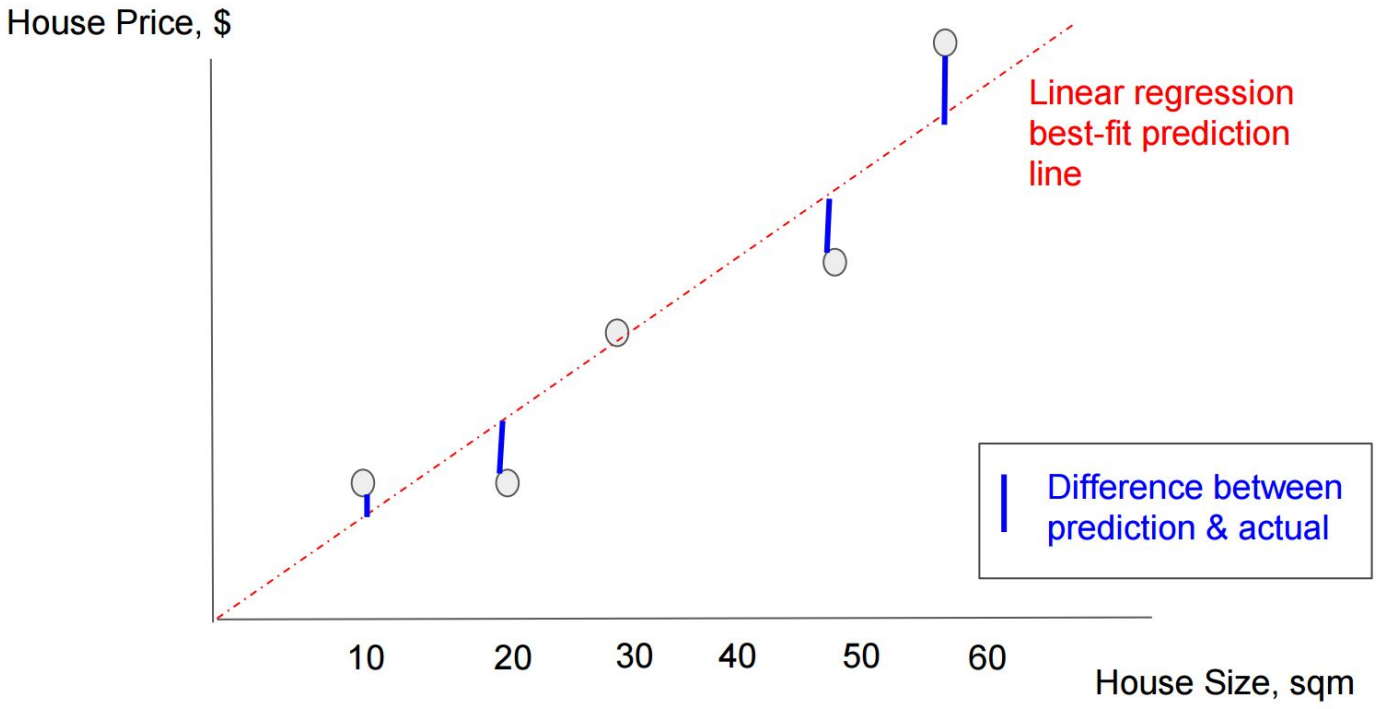
Cost Function: Best-fit Prediction

MLDS

Minimize difference between predicted & actual values

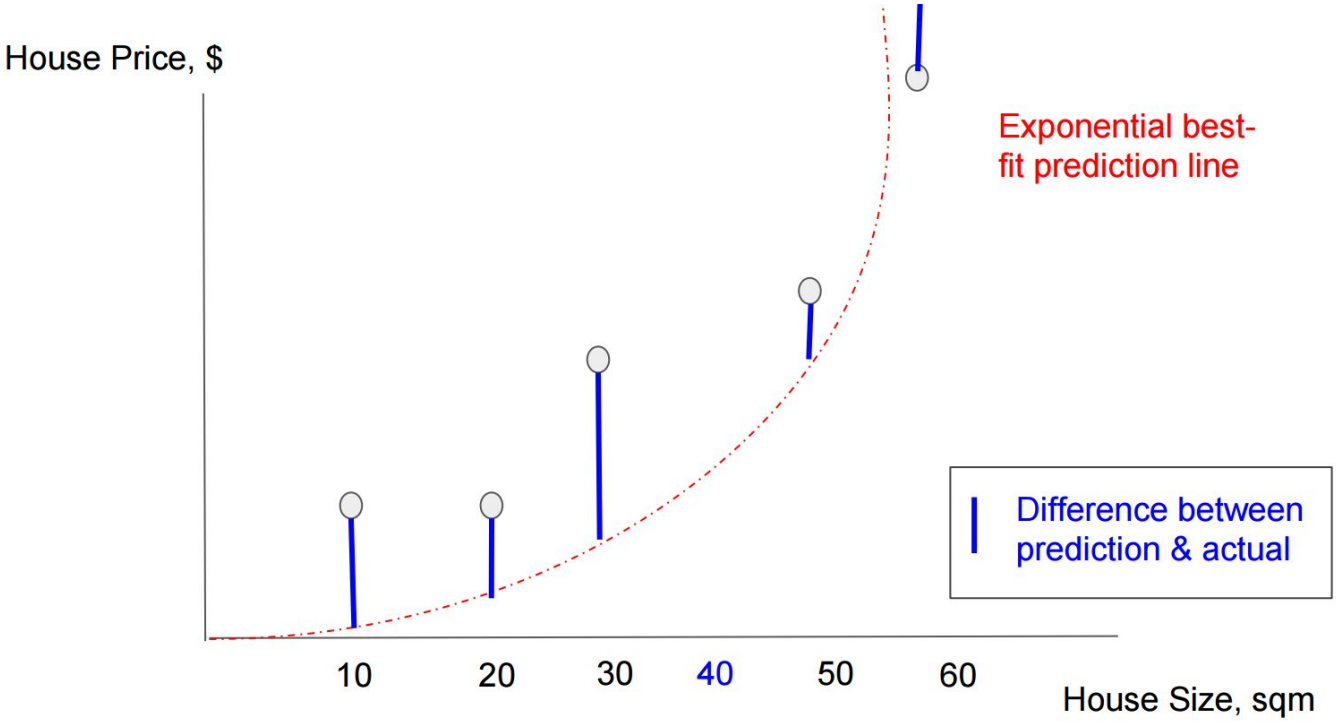
MLDS

What is the House Price?



MLDS

What is the House Price?



MLDS

Cost function: How different is your model from reality

sum((y_ - y)**2)

Where:

y_: Actual value

y: Predicted value

**2: Power of 2

MLDS

Linear Regression

y = W.x + b

y: Predicted house price

x: House size from dataset

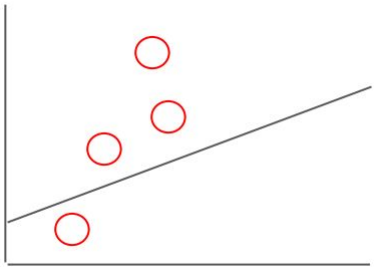
Find a good W, b

MLDS

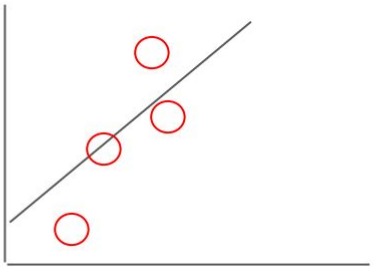
W: Gradient (Steepness)

W1 < W2 < W3

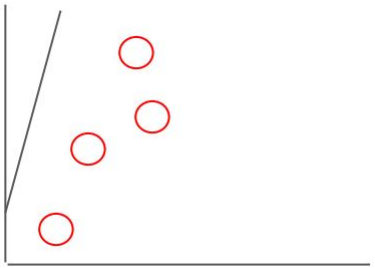
y = W1.x + b



y = W2.x + b



y = W3.x + b

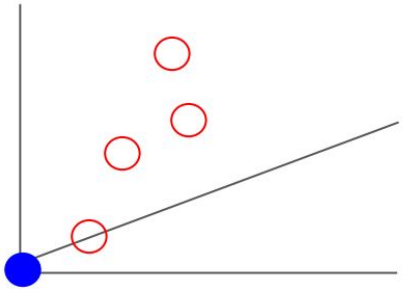


MLDS

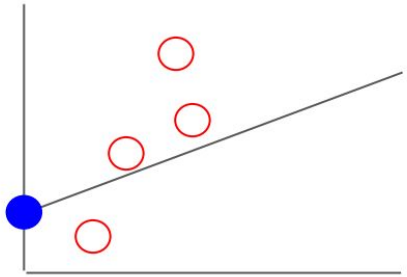
b: Intersect

b1 < b2 < b3

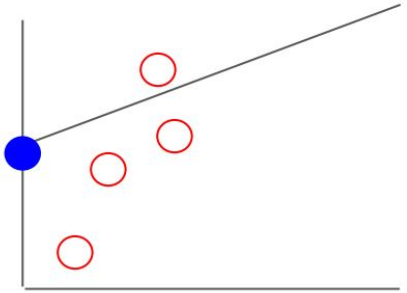
y = W.x + b1



y = W.x + b2



y = W.x + b3



MLDS

Train: Find W, b

W_best, b_best = 0, 0

Loop J times:

    Choose current_W, current_b

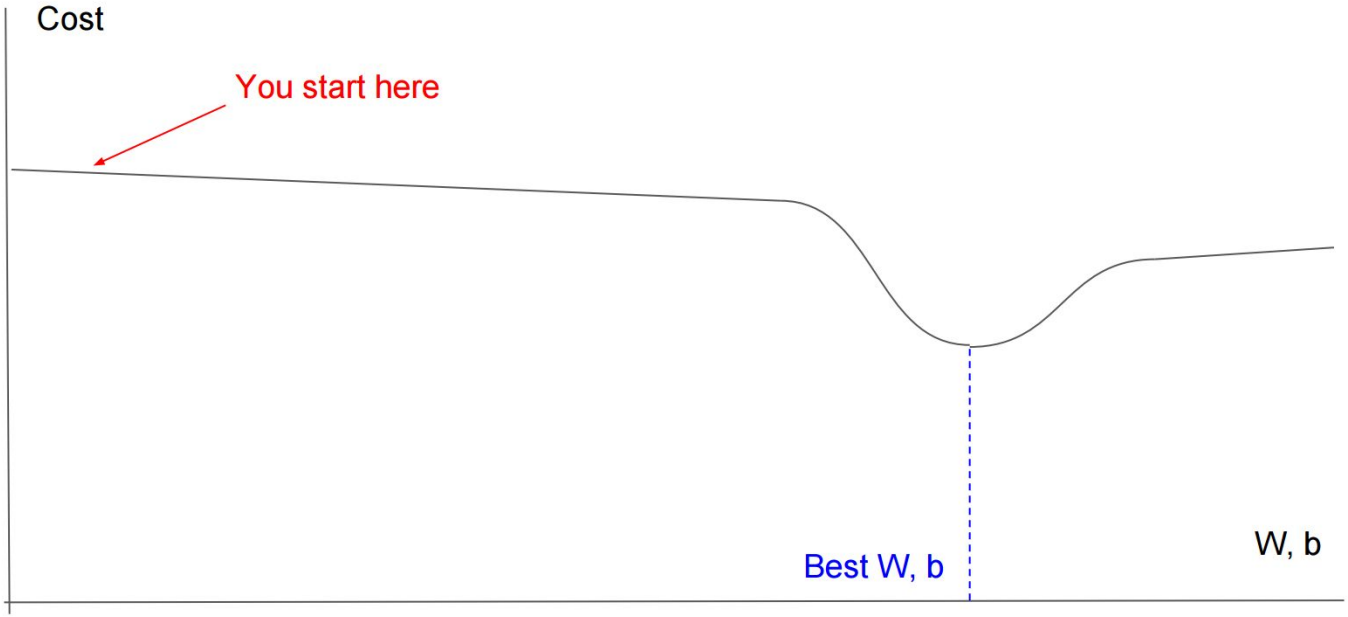
    If cost(current_W, current_b) < cost(W_best, b_best):

    W_best, b_best = current_W, current_b

What is the best way to choose current_W, current_b?

MLDS

Finding Minimum Cost Function



MLDS

Choosing Training Method

- Random
  - Could take forever

- Gradient descent
  - From current viewpoint, move towards direction where steepness is greatest

MLDS

Ready?

Recap:

- Model: Linear regression, y = W.x + b
- Cost: Least squared, cost = sum((y_ - y)**2)
- Train: Gradient descent

MLDS

Step 1: Model $\mathbf{y = W.x + b}$

MLDS

Tensorflow Model

Recap:

- Model: Linear regression, y = W.x + b
- Cost: Least squared, cost = sum((y_ - y)**2)
- Train: Gradient descent

MLDS

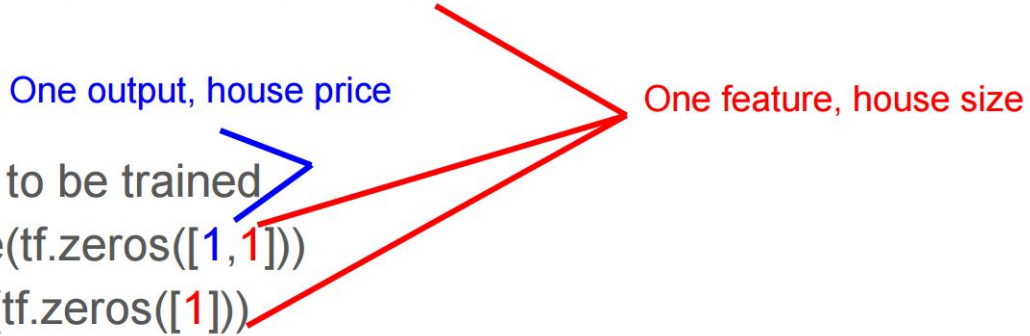
Tensorflow Model

tf.placeholder

- Hold data to be feed into the model
- x = tf.placeholder(tf.float32, [None, 1])

tf.Variable

- Hold variables to be trained
- W = tf.Variable(tf.zeros([1,1]))
- b = tf.Variable(tf.zeros([1]))



MLDS

Tensorflow Model

tf.matmul(x,W)

- Multiply 2 matrices
- product = tf.matmul(x,W)

‘+’

- y = product + b
- Expands to:
  - y = tf.matmul(x,W) + b = W.x + b

MLDS

Step 2: Cost Function

sum((y_ - y) 2)**

MLDS

Best-fit: Minimize Difference Prediction and Actual

Actual values:

One output, house price

- y_ = tf.placeholder(tf.float32, [None, 1])

Minimize difference between actual and prediction:

- cost = tf.reduce_sum(tf.pow((y_-y), 2))

MLDS

Step 3: Data

MLDS

Faking Data

```
for i in range(100):
  // Create fake data for actual data
  xs = np.array([[i]])
  ys = np.array([[2*i]])
```

- xs = 0, 1, 2, …., 99
- ys = 0, 2, 4, …., 198
- ys is always twice of xs
- **y = W.x + b where W = 2, b = 0 => Best fit: y = 2x**

MLDS

Step 4: Train

MLDS

Train using Gradient Descent

Train using Gradient Descent with steps in 0.000001

- train_step = tf.train.GradientDescentOptimizer(0.00001).minimize(cost)

MLDS

Train Model

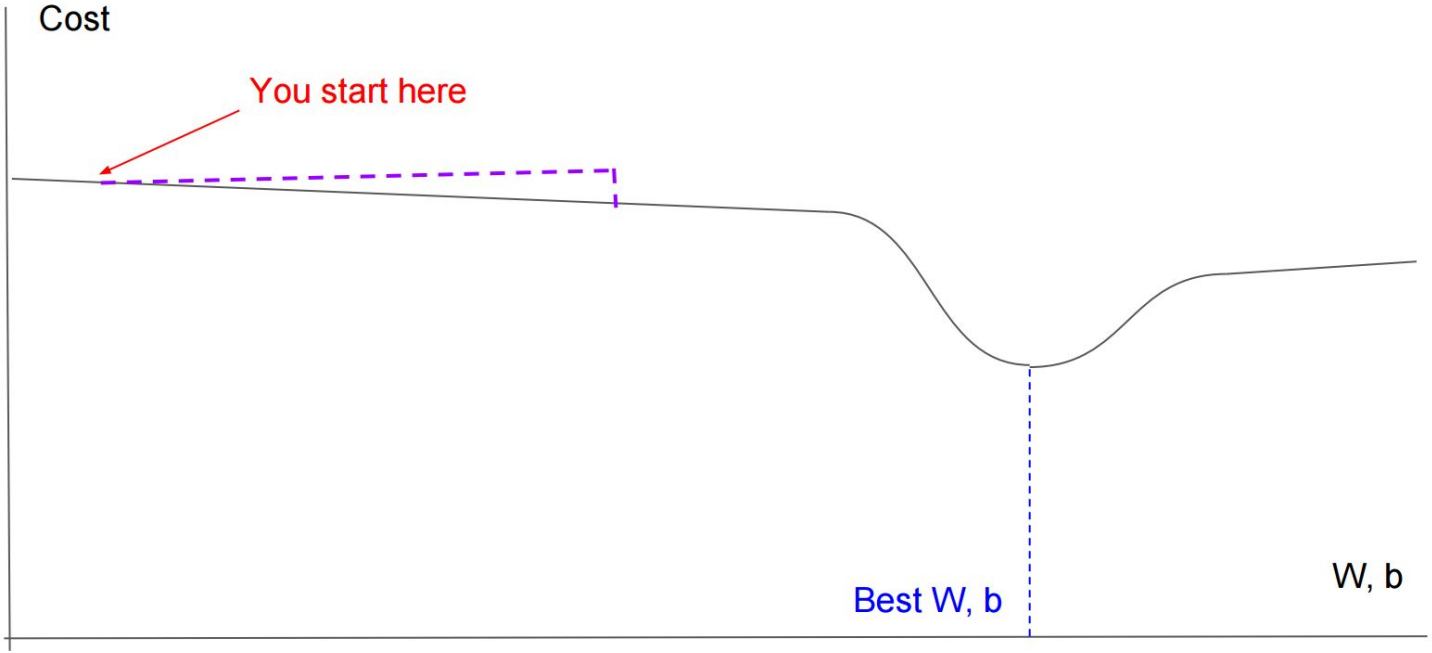
- sess = tf.Session()
- init = tf.initialize_all_variables()
- sess.run(init)
- steps = 100

Tensorflow Non-interactive Mode

Nothing is actually executed until sess.run(...)

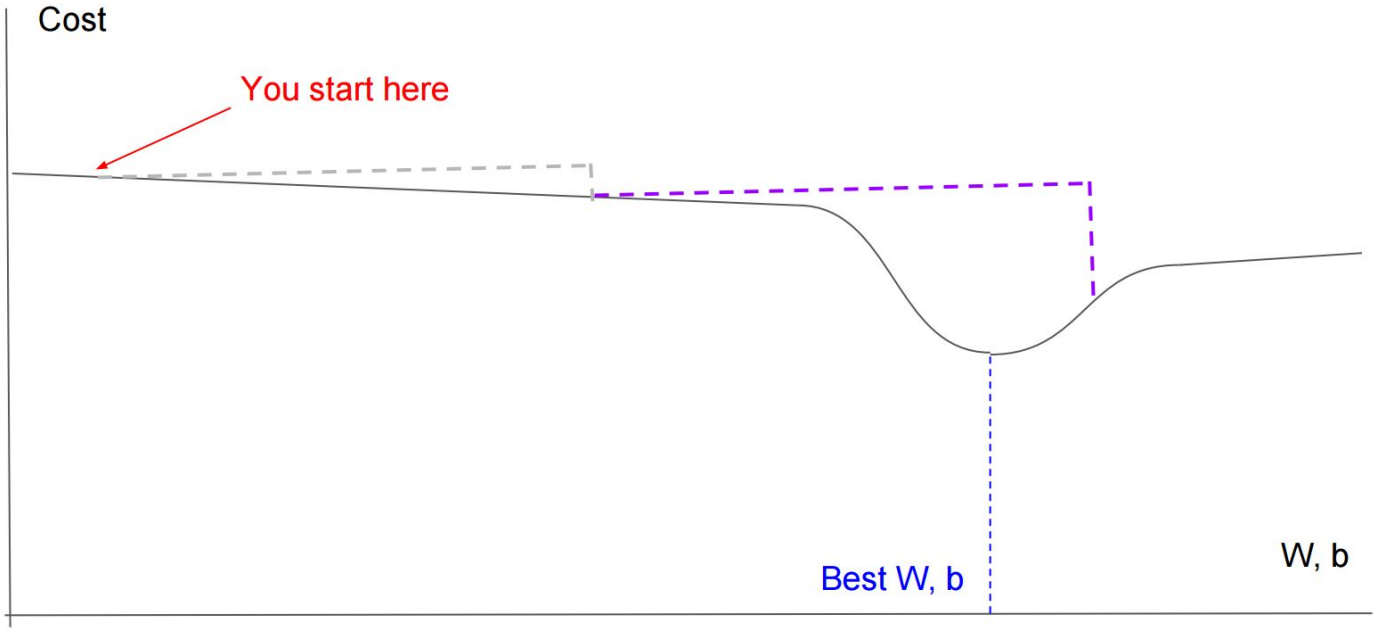
MLDS

Finding Minimum Cost Function



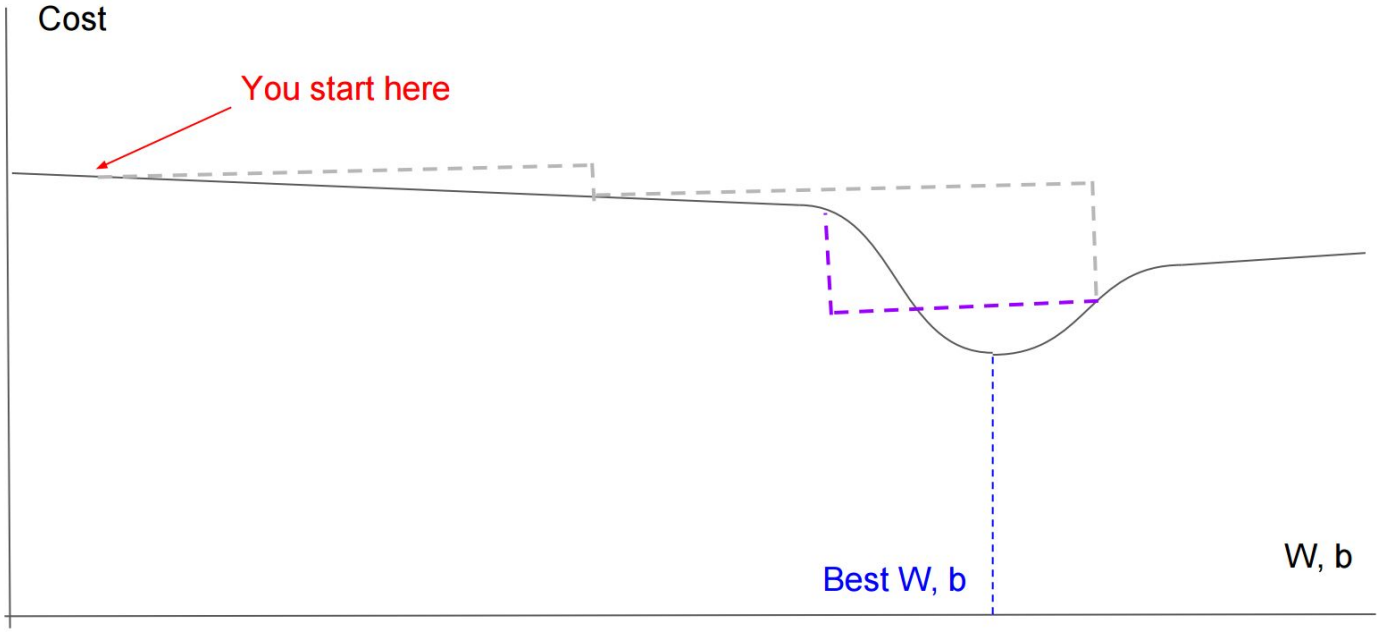
MLDS

Finding Minimum Cost Function



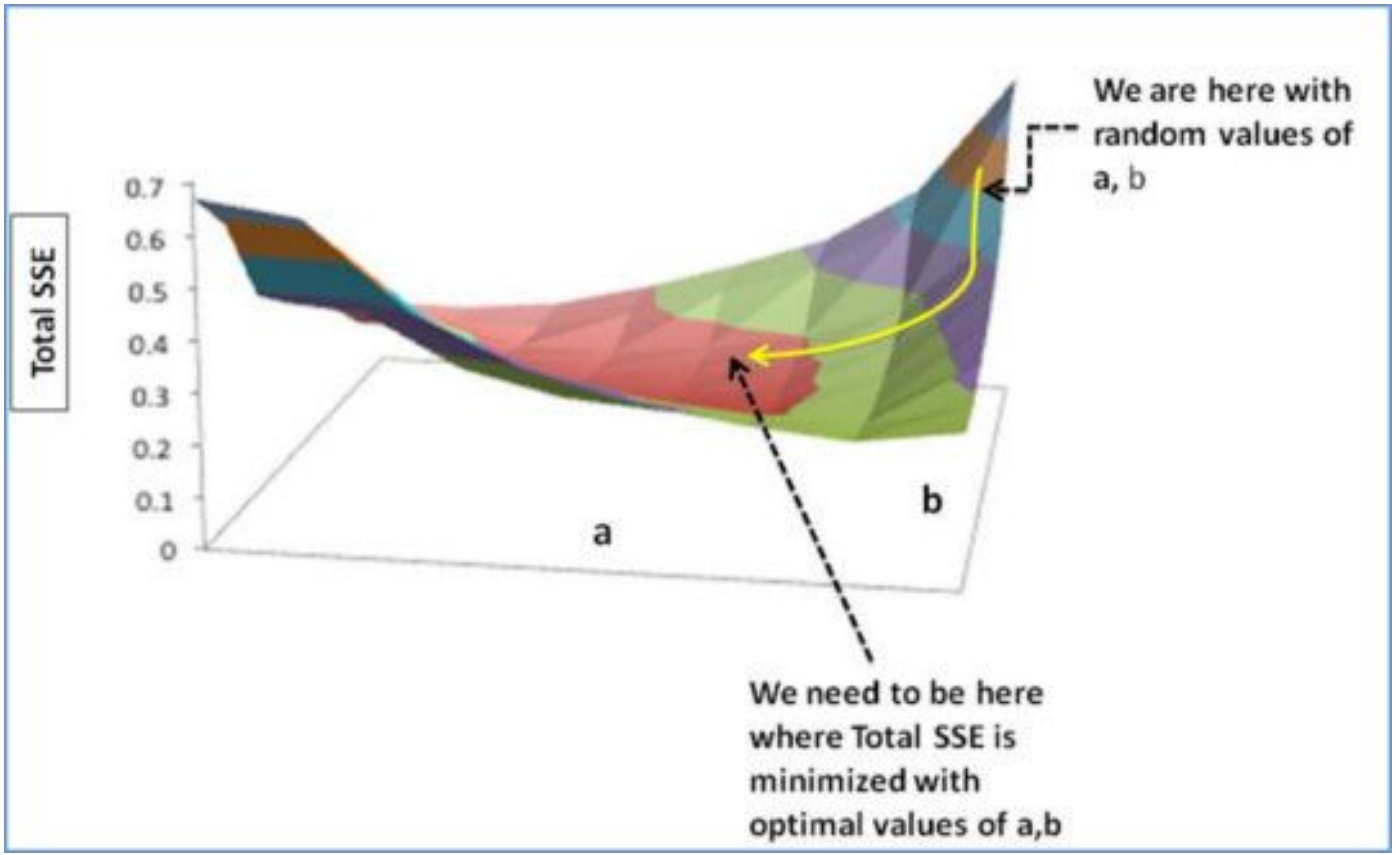
MLDS

Finding Minimum Cost Function



MLDS

Finding Minimum Cost Function



MLDS

Go!!!!!

```
for i in range(steps):
  # Create fake data for y = W.x + b where W = 2, b = 0
  xs = np.array([[i]])
  ys = np.array([[2*i]])

  # Train
  feed = { x: xs, y_: ys }
  sess.run(train_step, feed_dict=feed)

  print("After %d iteration:" % i)
  print("W: %f" % sess.run(W))
  print("b: %f" % sess.run(b))
```

MLDS

Thanks!

Machine Learning for Data Science Interest Group
Advanced Informatics School
Universiti Teknologi Malaysia

@utmmlds
ais.utm.my/mlds

May 2017