CMPE 258, Deep Learning

Feature scaling, overfitting

Jan 30, 2018

DMH 149A

Taehee Jeong

Ph.D., Data Scientist

SJSU SAN JOSÉ STATE UNIVERSITY

Office hour

Tuesday & Thursday, 5:45 – 6:30 PM

ENG 250

Walk-in or appointment

committed presences in classes

http://www.sjsu.edu/senate/docs/F15-3.pdf

Students should commit presence in classes, including the first week of weeks

Students should inform the instructor of the intention to continue in the class within the first 48 hours after the official class start date

Class instructors can drop students who fail to establish committed presences in classes within the 14th day of instruction (2/7)

HonestyPledge

23/40 is completed

Please sign honesty Pledge and upload the file.

There should be no except.

Please meet with me during office hour today if there is any reason you cannot upload it.

Prerequisites

18/40 is completed

CMPE 255 or CMPE 257

Please upload in Canvas a copy of your transcript, with the prerequisite class grade highlighted.

Please meet with me during office hour today if you did not complete the prerequisite classes.

Audit

The university has strict rules on auditing:
http://wiki.cmpe.sjsu.edu/doku.php/practices:no_unenrolled_students_in_class.

The students have to officially register class, select the audit option, and then they can stay.
Otherwise, no auditing is allowed.

Please let me know if you need permission number.

Course Schedule

Week	Date	Topics, Readings, Assignments, Deadlines
9	3/20	Convolutional Layer
9	3/22	Pooling Layer
10	3/27	Spring Recess
10	3/29	Spring Recess
11	4/3	CNN architectures
11	4/5	Midterm exam2
12	4/10	Recurrent Neural Networks
12	4/12	Memory cells, Input and Output Sequences
13	4/17	No class. Make up: (4/12, 5:45 – 7:00pm) Group Project proposals
13	4/19	No class. Make up: (4/24, 5:45 – 7:00pm) Group Project proposals
14	4/24	Training RNNs
14	4/26	LSTM, GRU
15	5/1	Autoencoders
15	5/3	Autoencoder application
16	5/8	Group Project Presentations
16	5/10	Group Project Presentations
Final Exam	5/16	Final Exam 2:45 pm – 5:00 pm

Group project

- Group Project Proposals, 4/12 & 4/24
- Group Project Presentations, 5/8 & 5/10

- Each Group should inform me which day prefer to present proposal & final presentation at least 2 weeks in advance with the name of members and the topic.

No class on 2/1

Make up

~~Option 1: start 12:00pm on 2/1~~

Option 2: start 4:00pm on 2/6 and 2/8

I will spent first 10-20min to recap previous lecture.

~~Option 3: No make up~~

Assignment_1

Any question?

1 (10pts). Linear regression with one variable
2 (30pts). Linear regression with two variables
3 (60pts). Linear regression with multiple variables

- from scratch (for loop, gradient descent)
- using matrix (gradient descent)
- using normal equation
- using scikit-learn linear regression model
- using TensorFlow gradient descent method

Recap

Linear regression

- Cost function
- Gradient descent
- Matrix calculation
- Normal equation

Linear regression

Cost function, Gradient descent

The objective of linear regression is to minimize the cost function

$$J = \frac{1}{m}\sum_{i=1}^{m}(\widehat{y^i} - y^i)^2$$

$$\hat{y} = b + Wx \; = \; W_0 + W_1 x$$

- *J : cost function*
- $\hat{y}: predicted\ target\ value$
- $y: actual\ target\ value$
- *x: input value*
- *b: bias*
- *W: weight*

Linear regression

(Batch) Gradient Descent

$$W_j := W_j - \alpha \frac{\partial J}{\partial W}$$

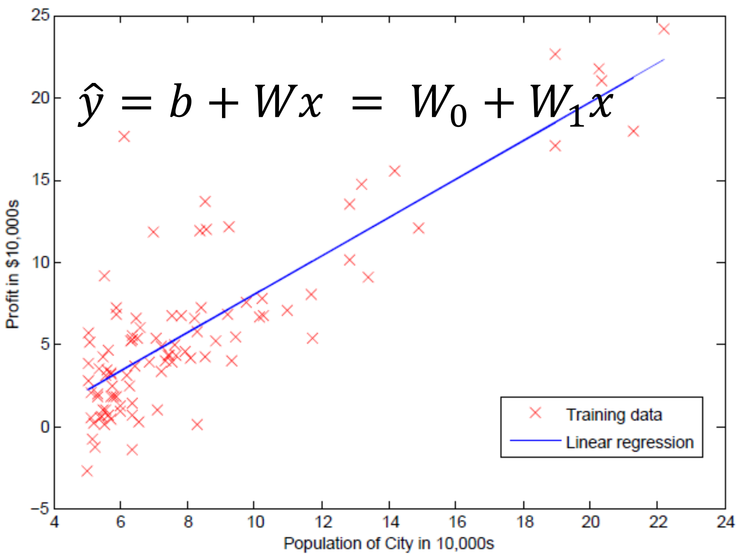
- α: $learning\ rate$
- $\frac{\partial J}{\partial W}$: gradient

$$\frac{\partial J}{\partial W_0} = \frac{2}{m} \sum_{i=1}^{m} (\widehat{y^i} - y^i)$$

$$\frac{\partial J}{\partial W_1} = \frac{2}{m} \sum_{i=1}^{m} (\widehat{y^i} - y^i)x$$

Linear regression

Linear regression fit



$$\hat{y} = b + Wx \;=\; W_0 + W_1 x$$

<Machine Learning, Andrew Ng>

- Weights
  $W_0, W_1, \ldots$
- Performance measure
  - Cost

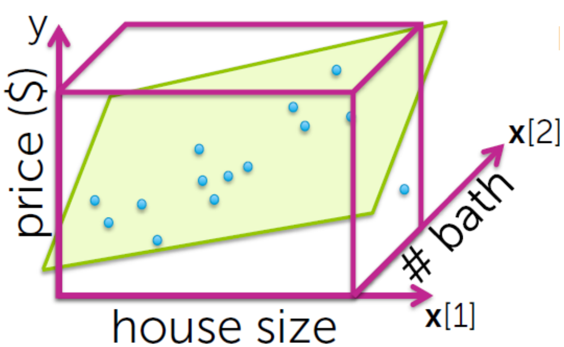
    $$J = \frac{1}{m}\sum_{i=1}^{m}(\widehat{y^i} - y^i)^2$$

  - Root Mean Square Error (RMSE)

    $$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\widehat{y^i} - y^i)^2}$$

Linear regression

With multiple variables



$$\hat{y} = W_0 + W_1 x_1 + W_2 x_2 + \ldots$$

$$\frac{\partial J}{\partial W_0} = \frac{2}{m}\sum_{i=1}^{m}(\widehat{y^i} - y^i)$$

$$\frac{\partial J}{\partial W_1} = \frac{2}{m}\sum_{i=1}^{m}(\widehat{y^i} - y^i)x_1$$

$$\frac{\partial J}{\partial W_2} = \frac{2}{m}\sum_{i=1}^{m}(\widehat{y^i} - y^i)x_2$$

<Machine Learning, Emily Fox & Carlos Guestrin>

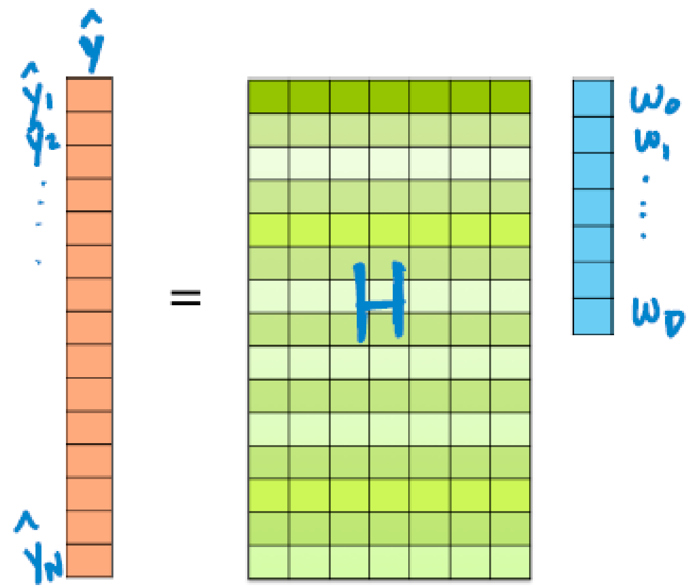
Linear regression

Matrix calculation for cost function

$$J = \frac{1}{m}\sum_{i=1}^{m}(\widehat{y^i} - y^i)^2$$

$$J = \frac{1}{m}(\hat{Y} - Y)^T(\hat{Y} - Y)$$

$$J = \frac{1}{m}(W \cdot X - Y)^T(W \cdot X - Y)$$



<Machine Learning, Emily Fox & Carlos Guestrin>

Linear regression

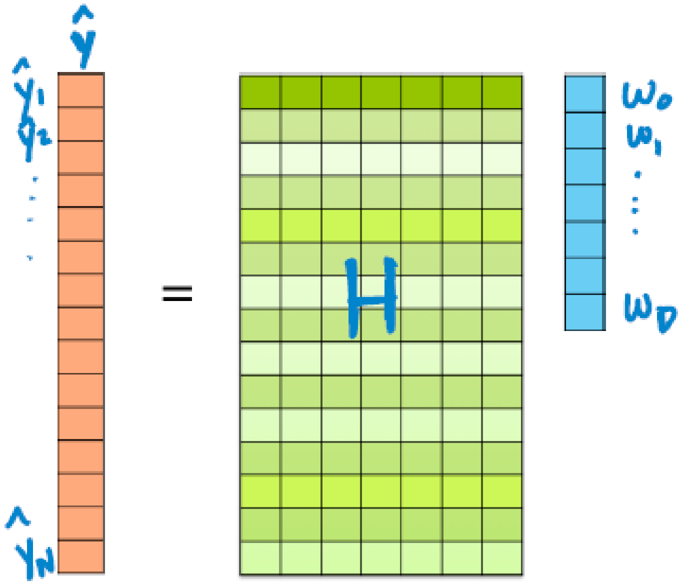
Matrix calculation for gradient

$$\frac{\partial J}{\partial W_j} = \frac{2}{m}\sum_{i=1}^{m}(\widehat{y^i} - y^i)x_j$$

$$\frac{\partial J}{\partial W} = \frac{2}{m}(\hat{Y} - Y)^T \cdot X$$

$$\frac{\partial J}{\partial W} = \frac{2}{m}(X \cdot W - Y)^T \cdot X$$

$$\frac{\partial J}{\partial W} = \frac{2}{m}(X^T \cdot X \cdot W - X^T \cdot Y)$$



<Machine Learning, Emily Fox & Carlos Guestrin>

Linear regression

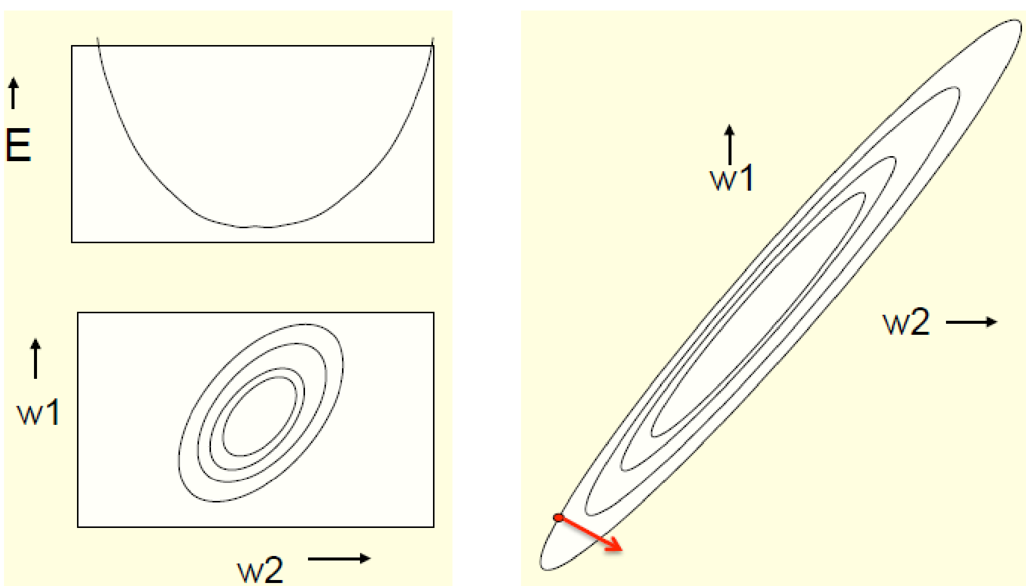
Normal equation

To find the value of θ that minimizes the cost function, there is a *closed-form solution*

—in other words, a mathematical equation that gives the result directly. This is called the *Normal Equation.*

$$W = (X^T \cdot X)^{-1} \cdot X^T \cdot Y$$

Feature scaling



- When the input numerical attributes have very different scales, Learning can be slow.
- If the ellipse is very elongated, the direction of steepest descent is almost perpendicular to the direction towards the minimum.

<Neural Networks, Geoffrey Hinton>

Feature Scaling

Feature normalization

- Min-Max scaling: (X – min.value) / (max.value – min.value)
  - shifted and rescaled. 0~1

- Standardization: (X – mean.value) / (std.value)
  - zero mean and unit standard deviation
  - does not bound a specific range.
  - less affected by outliers

Feature Scaling

Jupyter notebook

```
# Assignment_1 in Deep Learning, Spring 2018
```

```
#3. Linear regression with multiple variables
```

```
In [1]: import pandas as pd
```

```
In [4]: path = 'C:/san jose state university/Deep learning/'
        data3 = pd.read_csv(path + 'ex1data3.csv')
```

```
In [7]: del data3['Unnamed: 0']
```

```
In [8]: data3.shape
Out[8]: (20640, 9)
```

```
In [9]: data3.head()
Out[9]:
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	price
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Linear regression from scikit-learn

From sklearn.linear_model import LinearRegression

Lin_reg = LinearRegression()

Lin_reg.fit(X, y)

Lin_reg.predict(X)

Tensorflow

jupyter **Linear regression with tensorflow** Last Checkpoint: Last Friday at 9:53 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
In [1]: import tensorflow as tf

In [2]: x = tf.Variable(3, name = "x")
        y = tf.Variable(4, name = "y")
        f = x*x*y + y +2

In [3]: sess = tf.Session()

In [4]: sess.run(x.initializer)

In [5]: sess.run(y.initializer)

In [6]: result = sess.run(f)

In [7]: print(result)

        42

In [8]: sess.close()
```

1. Creates a session
2. Initializes the variables
3. Evaluates f
4. Closes the session (free up resources)

Tensorflow

with block

```
In [9]:  with tf.Session() as sess:
             x.initializer.run()
             y.initializer.run()
             result = f.eval()

In [10]: print (result)

42
```

Session is automatically closed at the end of the block

```
In [11]: init = tf.global_variables_initializer()
         with tf.Session() as sess:
             init.run()
             result = f.eval()

In [12]: print (result)

42
```

global_variables_initializer() function does not actually perform the initialization immediately, but rather creates a node in the graph that will initialized all variables when it is run.

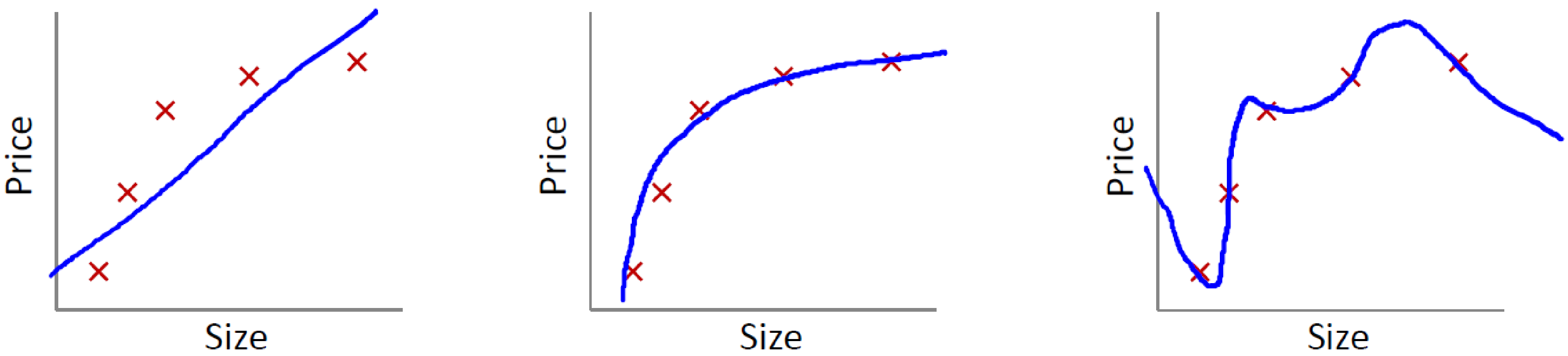
Tensorflow

Normal equation

In [14]:
```
import numpy as np
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()
m,n = housing.data.shape
housing_data_plus_bias = np.c_[np.ones((m,1)),housing.data]
X = tf.constant(housing_data_plus_bias, dtype = tf.float32, name  = "X")
y = tf.constant(housing.target.reshape(-1,1), dtype = tf.float32, name = "y")
XT = tf.transpose(X)
theta = tf.matmul(tf.matmul(tf.matrix_inverse(tf.matmul(XT,X)),XT),y)
```

In [15]:
```
with tf.Session() as sess:
    theta_value = theta.eval()
```

Overfitting/Underfitting



Overfitting: If we have too many features, the learned hypothesis
may fit the training set very well ( ), but fail to generalize to new examples

<Machine Learning, Andrew Ng>

Creating test set

70:30

60:20:20

Create a Test set

Using index

```
In [6]: import numpy as np

In [7]: def split_train_test(X, test_ratio):
            np.random.seed(1)
            shuffled_indices = np.random.permutation(len(X))
            test_set_size = int(len(X) * test_ratio)
            test_indices = shuffled_indices[:test_set_size]
            train_indices = shuffled_indices[test_set_size:]
            return X.iloc[train_indices], X.iloc[test_indices]

In [8]: train_set, test_set = split_train_test(data3, 0.2)

In [9]: train_set.shape

Out[9]: (16512, 9)

In [10]: test_set.shape

Out[10]: (4128, 9)
```

Create a Test set

Using sk_learn library

```
In [11]: from sklearn.model_selection import train_test_split

In [12]: train_set1, test_set1 = train_test_split(data3, test_size=0.2, random_state =1)

In [13]: train_set1.shape
Out[13]: (16512, 9)

In [14]: test_set1.shape
Out[14]: (4128, 9)
```