CMPE 258, Deep Learning

Regularization

Feb 6, 2018

DMH 149A

Taehee Jeong

Ph.D., Data Scientist

SJSU SAN JOSÉ STATE UNIVERSITY

Assignment_1

Due 2/4. 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

- Feature scaling (normalization)
- Linear regression from scikit-learn
- Tensorflow
- Overfitting & Underfitting
- Training / Validation/ Testing set
- Create a test data set

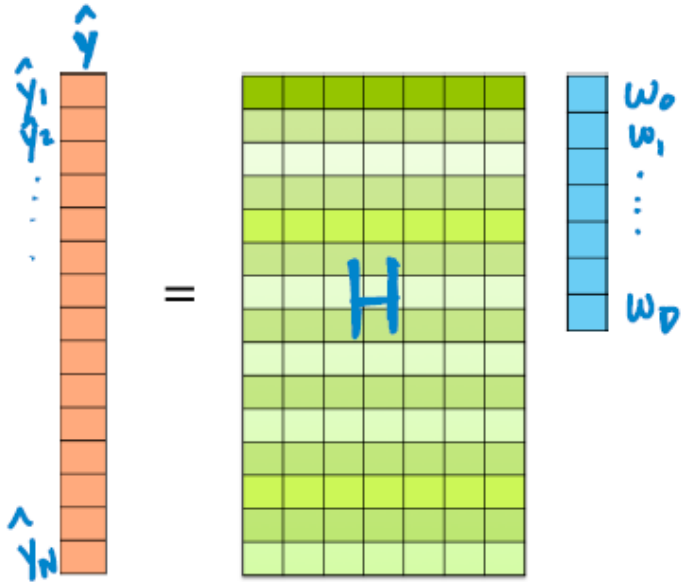
Linear regression

Matrix form 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

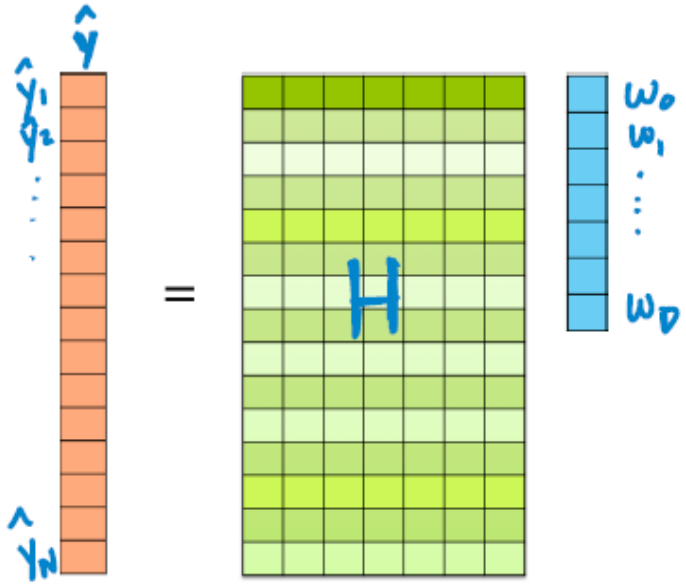
Matrix form for gradient

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

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

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

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



<Machine Learning, Emily Fox & Carlos Guestrin>

Learning Rate

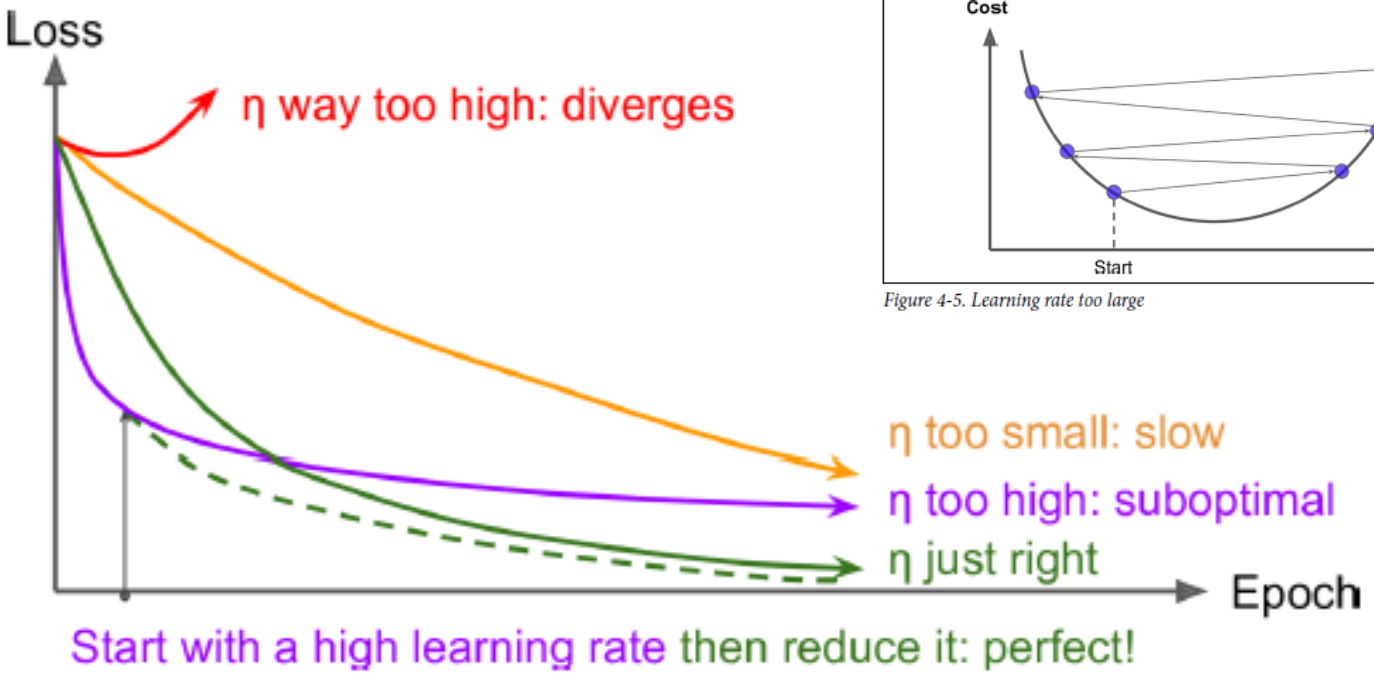


Figure 4-5. Learning rate too large

<Hands-On ML, Aurelien Geron>

Feature Scaling

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

Normal equation

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

Gradient descent

- Need to choose learning rate α
- Need to feature scaling
- Need to many iterations
- Works well even large number of features

Normal equation

- No Need to choose α
- No Need to feature scaling
- No need to iterations
- Needs to compute inverse matrix

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

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

Code

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

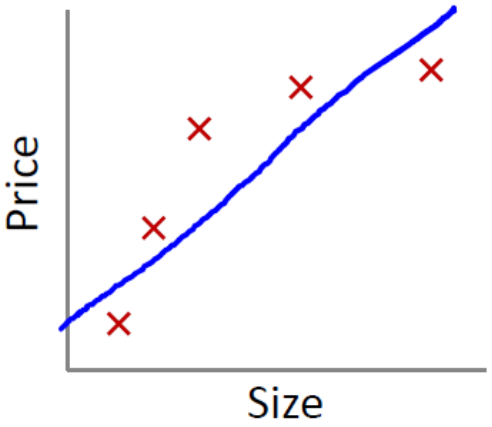
```
In [9]: with tf.Session() as sess:
            x.initializer.run()
            y.initializer.run()
            result = f.eval()

In [10]: print (result)
         42
```

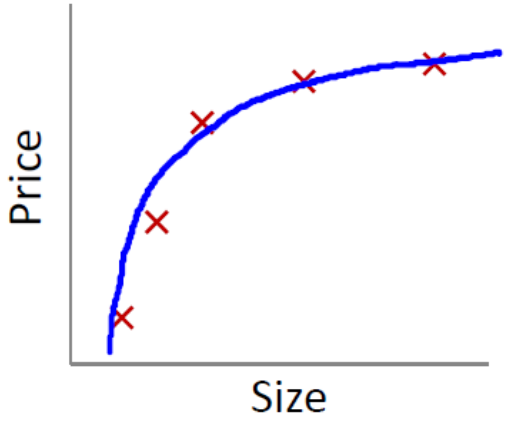
Session is automatically closed at the end of the block

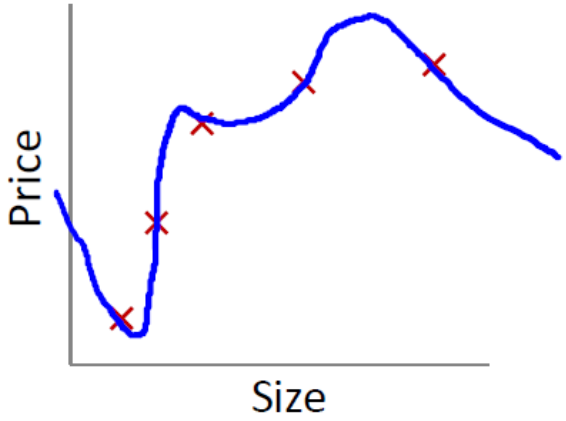
```
In [11]: init = tf.global_variables_initializer()
         with tf.Session() as sess:
             init.run()
             result = f.eval()

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.

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)
```

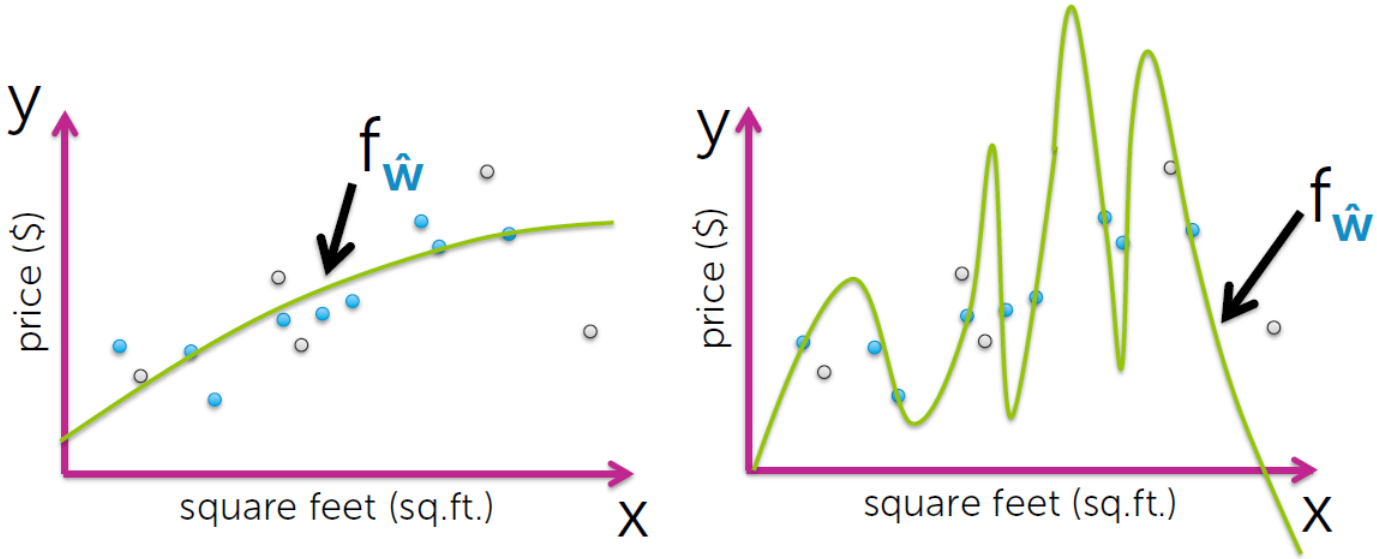
Feature normalization for test data

(X – mean.value) / (std.value)

In order to normalize validation / test data set, you need to use mean value and standard deviation value for training data set.

Overfitting of polynomial regression

$$\hat{y} = W_0 + W_1 x_1 + W_2 x_1^2 + W_3 x_1^3 + \cdots$$



<Machine Learning, Emily Fox & Carlos Guestrin>

Symptom of overfitting

- Very large value of regression coefficients, W
- Lots of input features
- Small number of observations

Avoiding Overfitting through Regularization

- **Sum of squares**

Ridge : L2 norm

$W_0^2 + W_1^2 + W_2^2 + \cdots = \sum_{j=1}^{n} W_j^2 = \|W\|_2^2$

Cost function $+\lambda\|W\|_2^2$

- **Sum of absolute value**

Lasso: L1 norm

$|w_0| + |w_1| + |w_2| + \cdots = \sum_{j=1}^{n} |w_j| = \|W\|_1$

Cost function $+\lambda\|W\|_1$

Regularized linear regression

Ridge regression (L2 penalty)

Cost function

$$J = \frac{1}{m}\sum_{i=1}^{m}(\widehat{y^i} - y^i)^2 + \frac{\lambda}{m}\sum_{j=1}^{n} W_j^2$$

Regularized linear regression

Ridge regression (L2 penalty)

Gradient

$$\frac{\partial J}{\partial W_0} = \frac{2}{m}\sum_{i=1}^{m}(\widehat{y^i} - y^i)x_0^i$$ when j=0

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

Regularized linear regression

Ridge regression (L2 penalty)

Gradient descant

Repeat {

$$W_0 = W_0 - \alpha \frac{2}{m} \sum_{i=1}^{m} (\widehat{y^i} - y^i) x_0^i \qquad \text{when j=0}$$

$$W_j = W_j - \alpha \left[\frac{2}{m} \sum_{i=1}^{m} (\widehat{y^i} - y^i) x_j^i + \frac{2\lambda}{m} W_j \right] \qquad \text{when j>=1}$$

}

Regularized linear regression

Ridge regression (L2 penalty)

Matrix form

Cost function

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

Gradient Descent

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

$$\frac{\partial J}{\partial W} = \frac{2}{m}[(X \cdot W - Y)^T \cdot X + \lambda I \cdot W]$$

I: identity matrix

Regularized linear regression

Ridge regression (L2 penalty)

Normalization equation for ridge regression

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

when λ=0, $W = (X^T \cdot X)^{-1} \cdot X^T \cdot Y$

When λ= infinite, W=0

Regularized linear regression

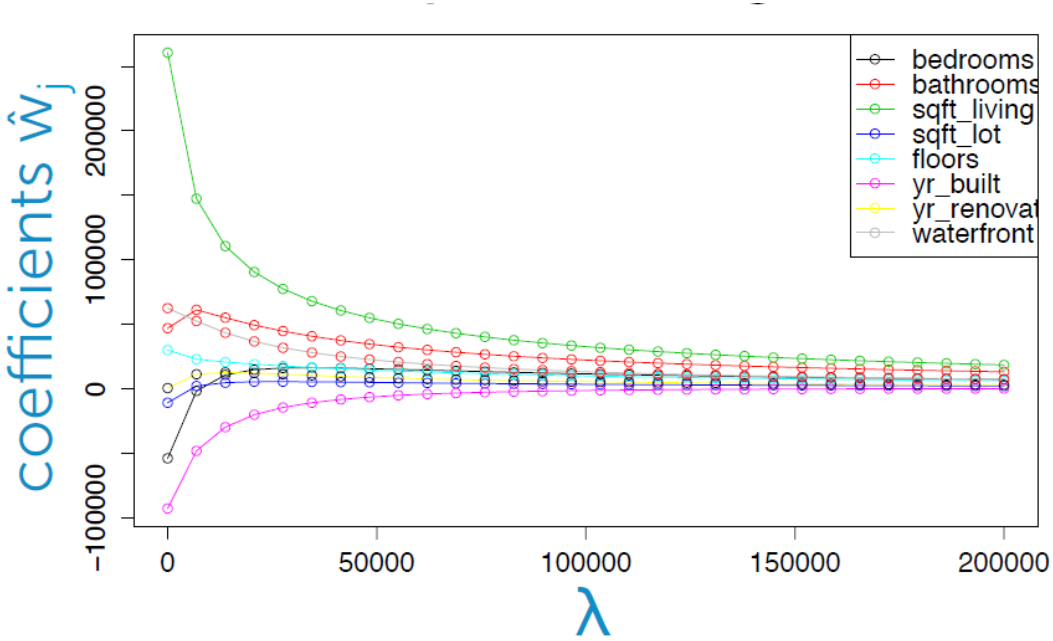
Ridge regression (L2 penalty)

Scikit-learn library

```
>>> from sklearn.linear_model import Ridge
>>> import numpy as np
>>> n_samples, n_features = 10, 5
>>> np.random.seed(0)
>>> y = np.random.randn(n_samples)
>>> X = np.random.randn(n_samples, n_features)
>>> clf = Ridge(alpha=1.0)
>>> clf.fit(X, y)
Ridge(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=None,
      normalize=False, random_state=None, solver='auto', tol=0.001)
```

Regularized linear regression

Ridge regression (L2 penalty)



<Machine Learning, Emily Fox & Carlos Guestrin>

Regularized linear regression

Lasso regression (L2 penalty)

Scikit-learn library

```
>>> from sklearn.linear_model import Ridge
>>> import numpy as np
>>> n_samples, n_features = 10, 5
>>> np.random.seed(0)
>>> y = np.random.randn(n_samples)
>>> X = np.random.randn(n_samples, n_features)
>>> clf = Ridge(alpha=1.0)
>>> clf.fit(X, y)
Ridge(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=None,
      normalize=False, random_state=None, solver='auto', tol=0.001)
```

<Scikit-learn.org>

Regularized linear regression

Lasso regression (L1 penalty)

Matrix form

Cost function

$$J = \frac{1}{m}[(W \cdot X - Y)^T (W \cdot X - Y) + \lambda |W|]$$

Gradient Descent

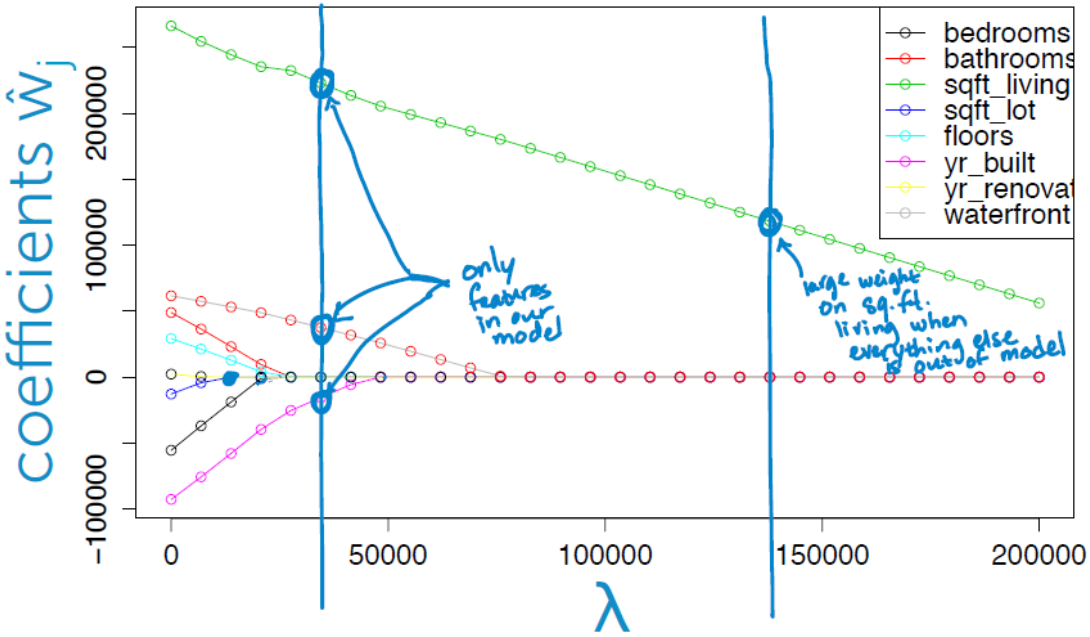
Coordinate descent method

<Machine Learning, Emily Fox & Carlos Guestrin>

$$A = \pi r^2$$

Regularized linear regression

Lasso regression (L1 penalty)



<Machine Learning, Emily Fox & Carlos Guestrin>

$A = \pi r^2$

Regularized linear regression

Lasso regression (L1 penalty)

Scikit-learn library

```
>>> from sklearn import linear_model
>>> clf = linear_model.Lasso(alpha=0.1)
>>> clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])
Lasso(alpha=0.1, copy_X=True, fit_intercept=True, max_iter=1000,
   normalize=False, positive=False, precompute=False, random_state=None,
   selection='cyclic', tol=0.0001, warm_start=False)
>>> print(clf.coef_)
[ 0.85  0.  ]
```

<Scikit-learn.org>

Bias / Variance Trade off

Large λ: high bias, low variance
Small λ: low bias, high variance

Bias

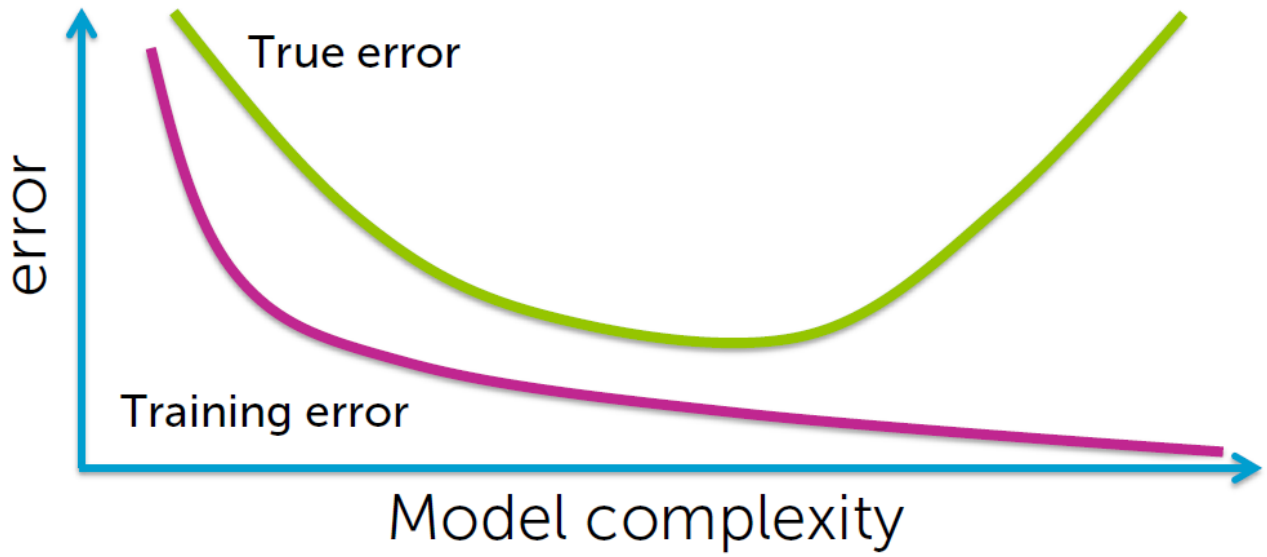
This part of the generalization error is due to wrong assumptions, such as assuming that the data is linear when it is actually quadratic. A high-bias model is most likely to underfit the training data.

Variance

This part is due to the model's excessive sensitivity to small variations in the training data. A model with many degrees of freedom (such as a high-degree polynomial model) is likely to have high variance, and thus to overfit the training data.

<Hands-On ML, Aurelien Geron>

Overfitting and regularization



<Machine Learning, Emily Fox & Carlos Guestrin>

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.

Course Schedule

Week	Date	Topics, Readings, Assignments, Deadlines
1	1/25	Introduction to Deep Learning, Linear regression
2	1/30	Cost function, Gradient Decent, Learning rate, Normal Equation
2	2/1	Overfitting, Underfitting, Training/Validating/Testing
3	2/6	Regularization, Bias-variance Trade-off
3	2/8	Logistic Regression, Binary classification
4	2/13	Softmax, Multiclass classification
4	2/15	Neural Networks , Hidden layers
5	2/20	Shallow Neural Networks, Backpropagation
5	2/22	Deep Neural Networks
6	2/27	Initialization: Xavier and He
6	3/1	Batch Normalization
7	3/6	Midterm exam1
7	3/8	Optimizers
8	3/13	Regularization
8	3/15	Convolutional Neural Networks

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.

Assignment_2

Due 2/15

1 (30pts). Polynomial regression / overfitting / regularization
2 (30pts). Polynomial regression with train/validation/test
3 (40pts). Regularization with Tensorflow

- using L2 penalty (Ridge)
- using L1 penalty (Lasso)
- using matrix (gradient descent)
- using scikit-learn linear regression model
- using TensorFlow gradient descent method

Summary

- Regularization:
  - sum of square value, L2 norm, Ridge
  - sum of absolute value, L1 norm, Lasso
  - Bias & Variance trade off
- 2/8
  - Mini-batch gradient descent
  - Regression with tensorflow