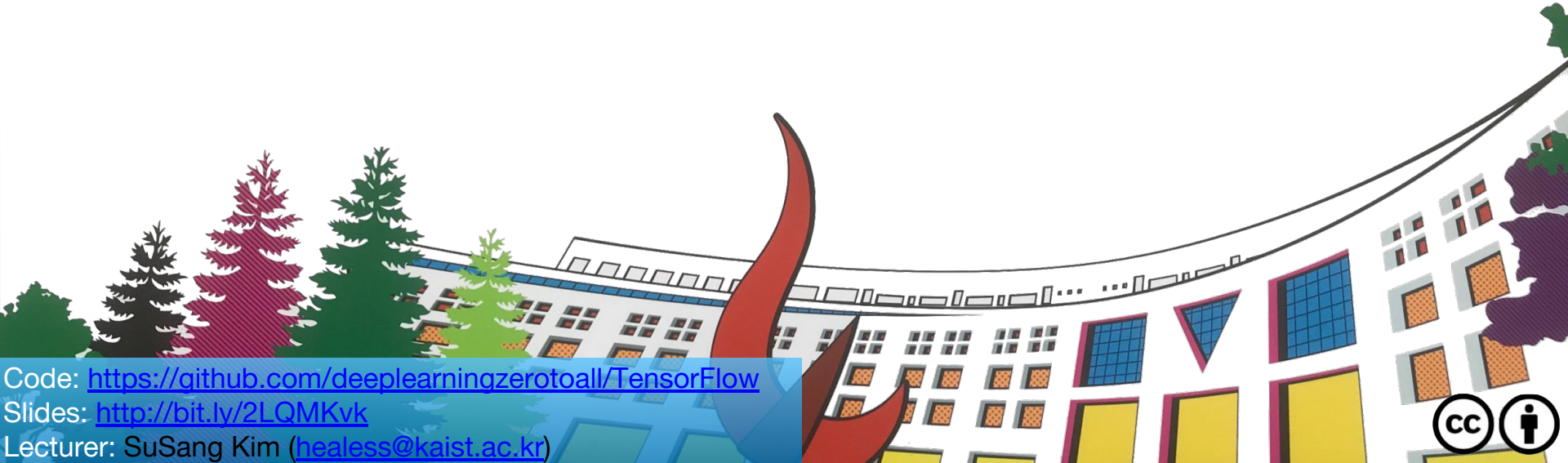
ML/DL for Everyone Season2

with TensorFlow

Lab 07-2 Application & Tips
Overfitting



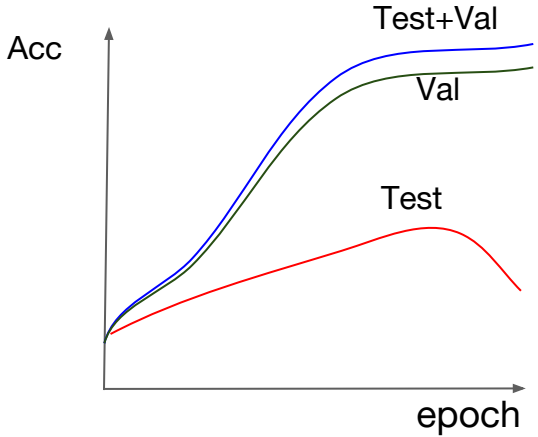
Code: https://github.com/deeplearningzerotoall/TensorFlow
Slides: http://bit.ly/2LQMKvk
Lecturer: SuSang Kim (healess@kaist.ac.kr)

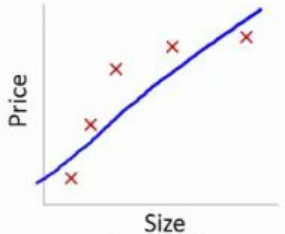
Application & Tips

- Learning rate
  - Gradient
  - Good and Bad Learning rate
  - Annealing the learning rate (Decay)
- Data preprocessing
  - Standardization / Normanalization
  - Noisy Data
- Overfitting
  - Set a features
  - Regularization
- Codes(Eager Execution)
- Summary

Andrew Ng's ML Class http://www.holehouse.org/mlclass/

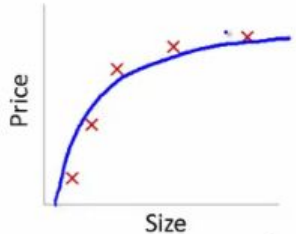
Overfitting





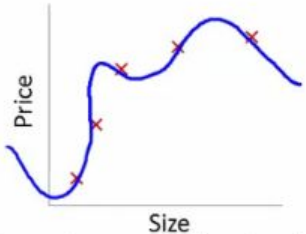
$\theta_0 + \theta_1 x$

High bias
(underfit)



$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance
(overfit)

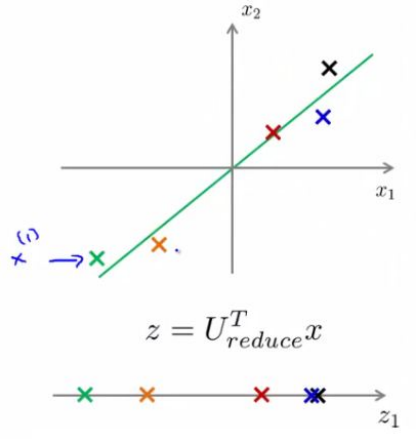
FaceScrub dataset(Aaron Eckhart)

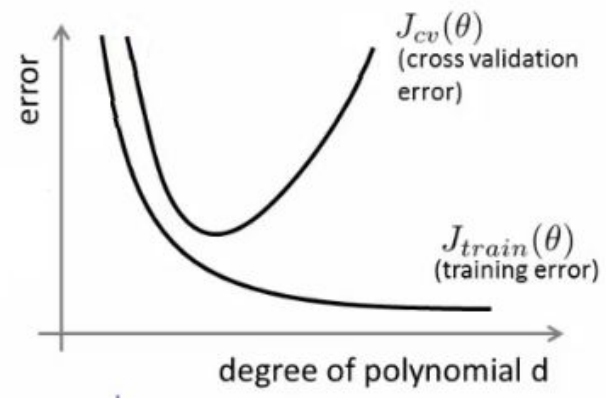
http://www.holehouse.org/mlclass/10_Advice_for_applying_machine_learning.html

Overfitting

Set a features

- Get more training data - more data will actually make a difference, (helps to fix high variance)
- Smaller set of features - dimensionality reduction(PCA) (fixes high variance)
- Add additional features - hypothesis is too simple, make hypothesis more specific (fixes high bias)





1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$

$\vdots$

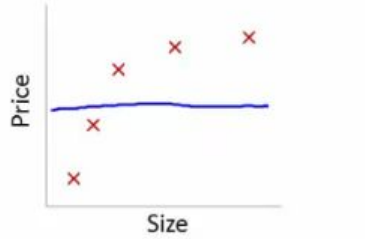
10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$

[sklearn Code]

```
from sklearn.decomposition import PCA
pca = decomposition.PCA(n_components=3)
pca.fit(X)
X = pca.transform(X)
```

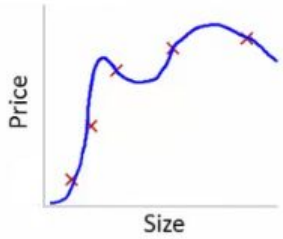
Overfitting

Regularization (Add term to loss)



Large λ
High bias (underfit)
$\lambda = 10000.\ \theta_1 \approx 0, \theta_2 \approx 0, \ldots$
$h_\theta(x) \approx \theta_0$

Intermediate λ
"Just right"



Small λ
High variance (overfit)
$\lambda \approx 0$

λ-- : fixes high bias (Under fitting)
λ ++ : fixes high variance (overfitting)

Linear regression with regularization

Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^{m} \theta_j^2$$

[Tensorflow Code]

```
L2_loss = tf.nn.l2_loss(w) # output = sum(t ** 2) / 2
```

Overfitting

Solutions

- Feature Normalization
- Regularization
- More Data and Data Augmentation
  - Color Jiltering
  - Horizontal Flips
  - Random Crops/Scales
- Dropout (0.5 is common)
- Batch Normalization

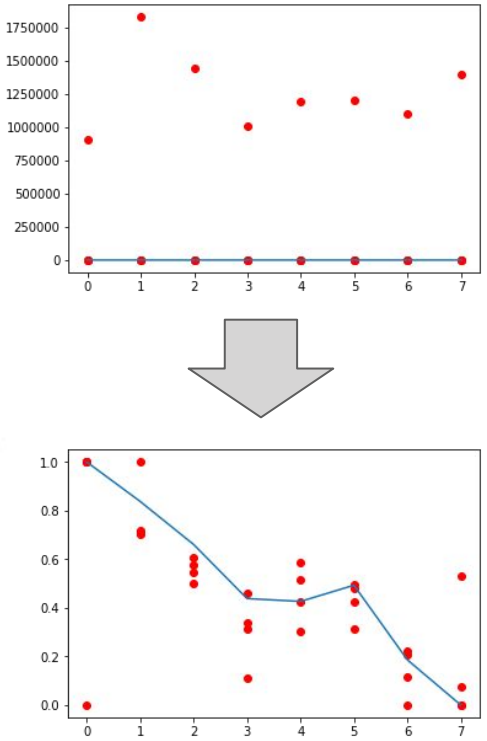
Code(Eager)

Data Preprocess

```
import tensorflow.contrib.eager as tfe
tf.enable_eager_execution()

xy = np.array([[828.659973, 833.450012, 908100, 828.349976, 831.659973],
               [823.02002, 828.070007, 1828100, 821.655029, 828.070007],
               [819.929993, 824.400024, 1438100, 818.97998, 824.159973],
               [816, 820.958984, 1008100, 815.48999, 819.23999],
               [819.359985, 823, 1188100, 818.469971, 818.97998],
               [819, 823, 1198100, 816, 820.450012],
               [811.700012, 815.25, 1098100, 809.780029, 813.669983],
               [809.51001, 816.659973, 1398100, 804.539978, 809.559998]])

x_train = xy[:, 0:-1]
y_train = xy[:, [-1]]

def normalization(data):
    numerator = data - np.min(data, 0)
    denominator = np.max(data, 0) - np.min(data, 0)
    return numerator / denominator

xy = normalization(xy)
```

Normalization
(0~1)

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

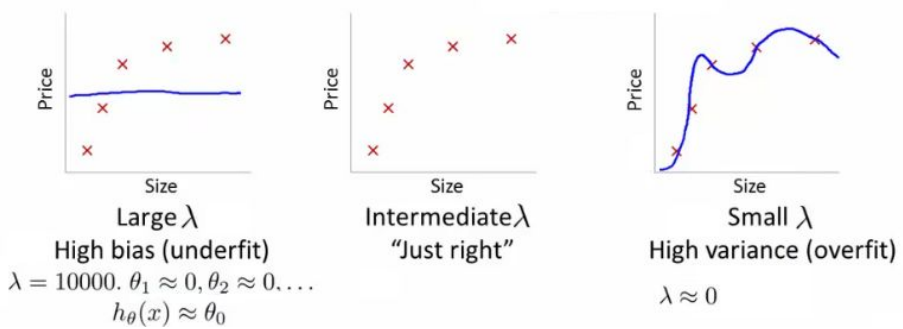


https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-07-3-linear_regression_eager%20(preprocess%2C%20decay%2C%20l2_loss).ipynb

Code(Eager)

L2 Norm

```
dataset = tf.data.Dataset.from_tensor_slices((x_train, y_train)).batch(len(x_train))

W = tf.Variable(tf.random_normal([4, 1]), dtype=tf.float32)
b = tf.Variable(tf.random_normal([1]), dtype=tf.float32)

def linearReg_fn(features):
    hypothesis = tf.matmul(features, W) + b
    return hypothesis

def l2_loss(loss, beta = 0.01):
    W_reg = tf.nn.l2_loss(W) # output = sum(t ** 2) / 2
    loss = tf.reduce_mean(loss + W_reg * beta)
    return loss

def loss_fn(hypothesis, labels, flag = False):
    cost = tf.reduce_mean(tf.square(hypothesis - labels))
    if(flag):
        cost = l2_loss(cost)
    return cost
```



Linear regression with regularization

Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-07-3-linear_regression_eager%20(preprocess%2C%20decay%2C%20l2_loss).ipynb

Code(Eager)

Learning Decay

```
is_decay = True
starter_learning_rate = 0.1

if(is_decay):
   global_step = tf.Variable(0, trainable=False)
   learning_rate = tf.train.exponential_decay(starter_learning_rate, global_step, 50, 0.96, staircase=True)
   optimizer = tf.train.GradientDescentOptimizer(learning_rate)
else:
   optimizer = tf.train.GradientDescentOptimizer(learning_rate=starter_learning_rate)

def grad(features, labels, l2_flag):
   with tf.GradientTape() as tape:
       loss_value = loss_fn(linearReg_fn(features),labels, l2_flag)
   return tape.gradient(loss_value, [W,b]), loss_value

for step in range(EPOCHS):
   for features, labels  in tfe.Iterator(dataset):
       features = tf.cast(features, tf.float32)
       labels = tf.cast(labels, tf.float32)
       grads, loss_value = grad(linearReg_fn(features), features, labels, False)
       optimizer.apply_gradients(grads_and_vars=zip(grads,[W,b]), global_step=global_step)
       if step % 10 == 0:
           print("Iter: {}, Loss: {:.4f}, Learning Rate: {:.8f}".format(step, loss_value,
optimizer._learning_rate()))
```

```
Iter: 0, Loss: 1.7346, Learning Rate: 0.10000000
Iter: 10, Loss: 0.0745, Learning Rate: 0.10000000
Iter: 20, Loss: 0.0438, Learning Rate: 0.10000000
Iter: 30, Loss: 0.0273, Learning Rate: 0.10000000
Iter: 40, Loss: 0.0181, Learning Rate: 0.10000000
Iter: 50, Loss: 0.0128, Learning Rate: 0.09600000
Iter: 60, Loss: 0.0099, Learning Rate: 0.09600000
Iter: 70, Loss: 0.0080, Learning Rate: 0.09600000
Iter: 80, Loss: 0.0068, Learning Rate: 0.09600000
Iter: 90, Loss: 0.0060, Learning Rate: 0.09600000
Iter: 100, Loss: 0.0054, Learning Rate: 0.09216000
```

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-07-3-linear_regression_eager%20(preprocess%2C%20decay%2C%20l2_loss).ipynb

Summary

- **Learning rate**
  - Gradient
  - Good and Bad learning rate
  - Annealing the learning rate (Decay)
- **Data preprocessing**
  - Standardization / Normanalization
  - Noisy Data
- **Overfitting**
  - Set a Features
  - Regularization