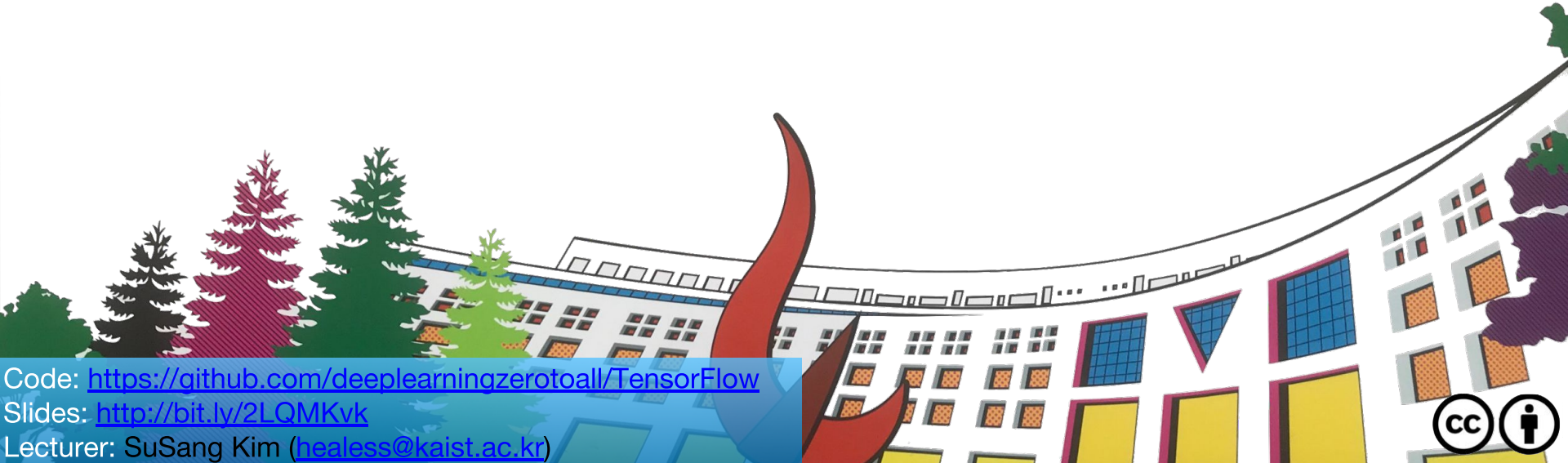
ML/DL for Everyone Season2

with TensorFlow

Lab 07-1 Application & Tips



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

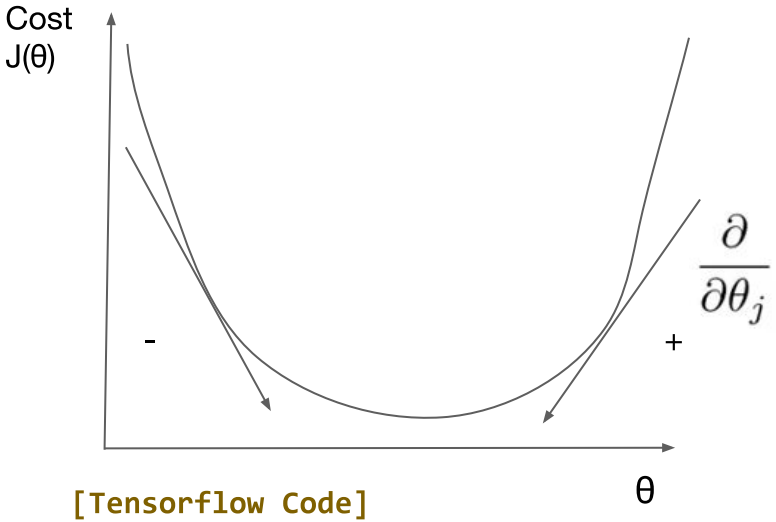
Application & Tips

- Learning rate
  - Gradient
  - Good and Bad Learning rate
  - Annealing the learning rate (Decay)
- Data preprocessing
  - Standardization / Normanalization
  - Noisy Data
- Overfitting
  - Regularization
  - L2 Norm

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

Learning rate

Gradient



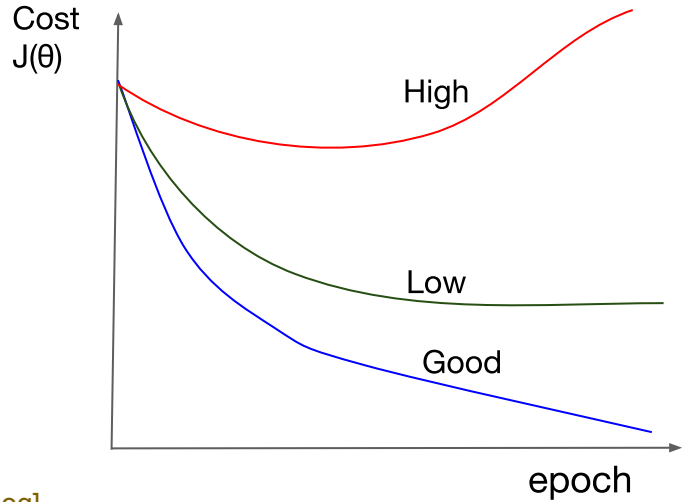
Repeat $\{ \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \}$

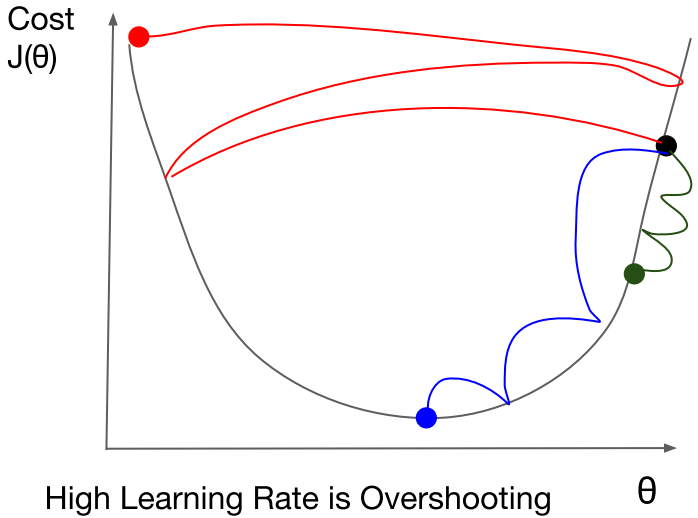
Learning rate is a hyper-parameter that controls how much we are adjusting the weights with respect the loss gradient

```
[Tensorflow Code]
def grad(hypothesis, labels):
   with tf.GradientTape() as tape:
       loss_value = loss_fn(hypothesis, labels)
   return tape.gradient(loss_value, [W,b])
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
optimizer.apply_gradients(grads_and_vars=zip(grads,[W,b]))
```

Learning rate

Good and Bad





High Learning Rate is Overshooting
Normal Learning Rate is `0.01`
`3e-4` is the best learning rate for Adam, hands down (andrej karpathy)

[Train Log]

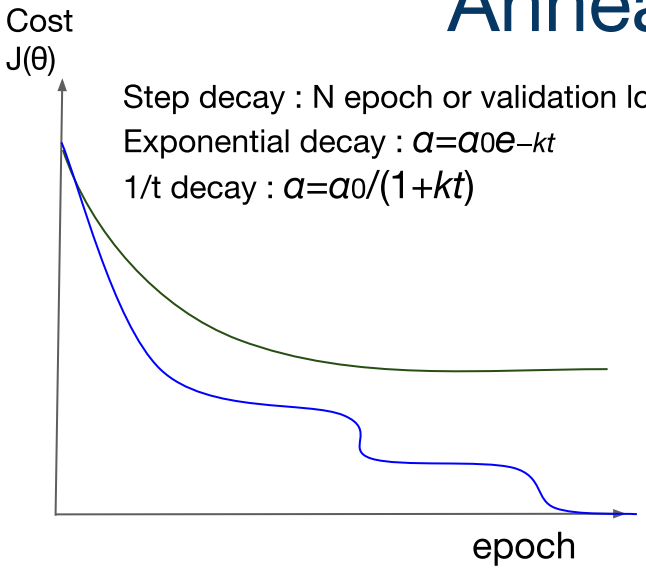
```
Iter: 0, Loss: 6.0257, Learning Rate: 0.1000
Iter: 1000, Loss: 0.3723, Learning Rate: 0.0960
Iter: 2000, Loss: 0.2779, Learning Rate: 0.0922
Iter: 3000, Loss: 0.2293, Learning Rate: 0.0885
Iter: 4000, Loss: 0.1977, Learning Rate: 0.0849
Iter: 5000, Loss: 0.1750, Learning Rate: 0.0815
```

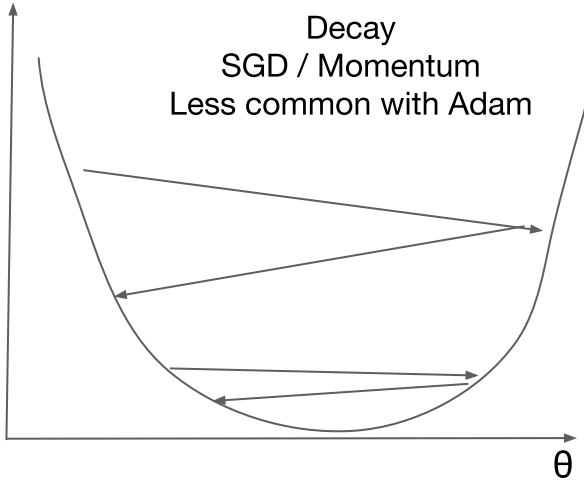
[Tensorflow Code]

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
```

Learning rate

Annealing the learning rate





Log

```
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: 0.0921600
```

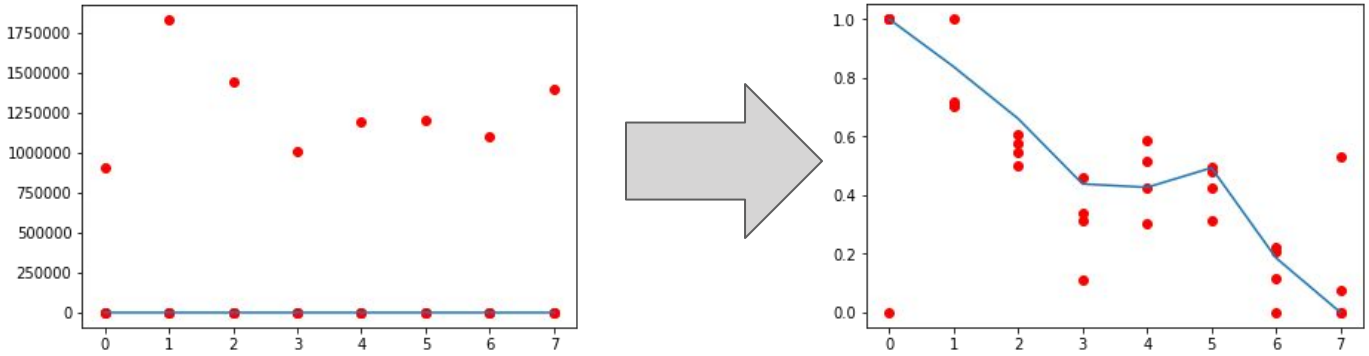
[Tensorflow Code]

```
learning_rate = tf.train.exponential_decay(starter_learning_rate,
                global_step, 1000, 0.96, staircase=True)
# tf.train.exponential_decay / tf.train.inverse_time_decay
# tf.train.natural_exp_decay / tf.train.piecewise_constant
# tf.train.polynomial_decay
```

```
def exponential_decay(epoch):
  starter_rate = 0.01
  k = 0.96
  exp_rate = starter_rate  * exp(-k*t)
  return exp_rate
```

Data preprocessing

Feature Scaling



Standardization
(Mean Distance)

$$x_{new} = \frac{x - \mu}{\sigma}$$

```
[Python Code(numpy)]
Standardization = (data - np.mean(data)) / sqrt(np.sum(
(data - np.mean(data))^2 ) / np.count(data))
```

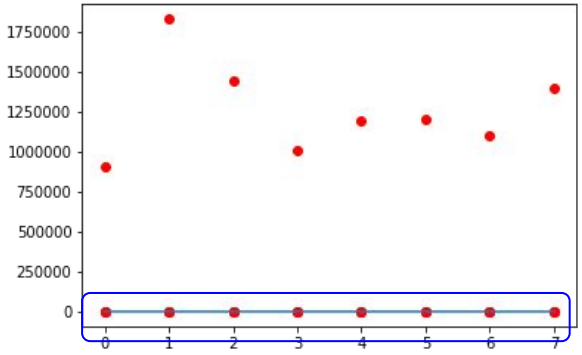
Normalization
(0~1)

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

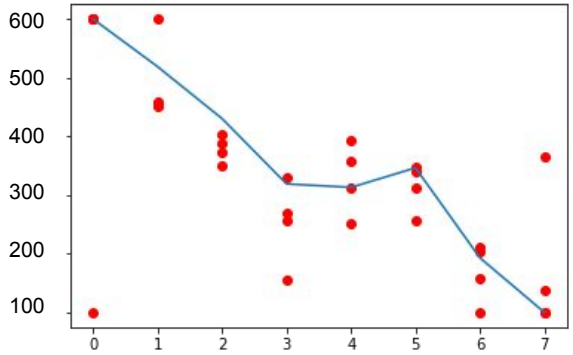
```
Normalization = (data - np.min(data, 0)) / (np.max(data, 0)
- np.min(data, 0))
```

Data preprocessing

Noisy Data



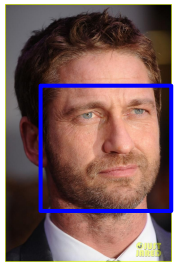
Numeric



Will you order a pizza..??

NLP

You order pizza



Face Image

FaceScrub dataset(Aaron Eckhart)