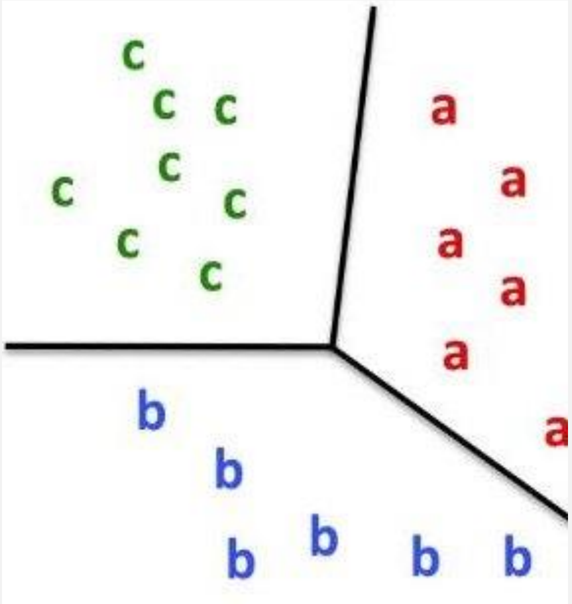
Google Developer Student Clubs

ML/DL Study W02

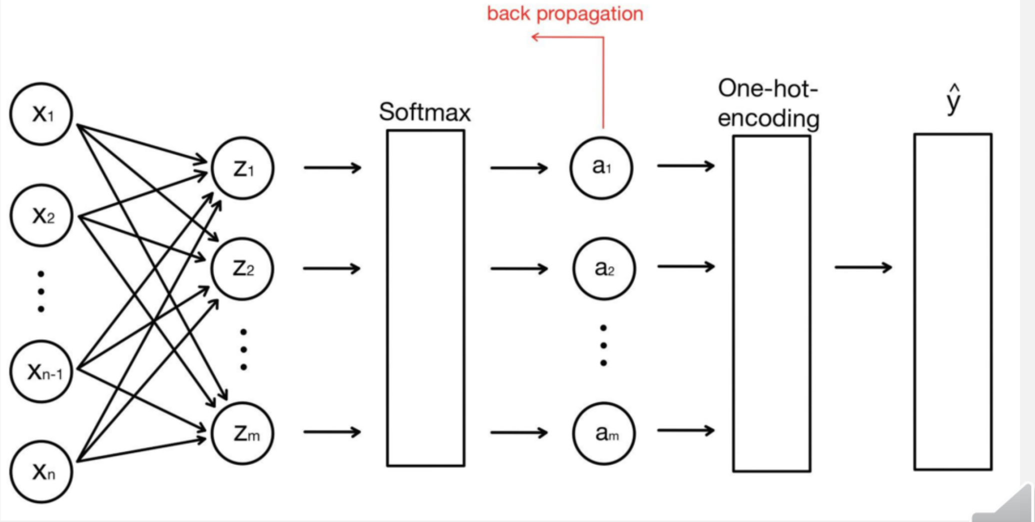
- Soft max
- Cross – entropy
- Learning Rate
- Data Preprocessing
- Overfitting
- Solution

Soft max

Soft max



Multinomial classification



Softmax : input x를 [0:1] 사이의 값으로 모두 정규화하며 출력시키고, 출력 값의 총합은 항상 1이 됨

$$y = [\,2.0,\ 1.0,\ 0.1\,]$$

$$\boxed{S(y)}$$

$$\bar{y} = [\,0.7,\ 0.2,\ 0.1\,]$$

Cross – entropy

Cross – entropy

모델에서 예측한 확률값이 실제값과 비교했을 때 틀릴 수 있는 정보량

Softmax (y) = ȳ

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

Cross - entropy

$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

Logistic cost

i = [1 : m] , training data set

Cross – entropy

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

1) 실제값(y, L)이 0이고, 예측값(H(x), S)이 0인 경우
Logistic Cost -> -(1-0)log(1-0) = -log(1) = 0
Cross Entropy -> Sum [1 // 0] * [-log(1) // -log(0)] = Sum [0 // 0] = 0

2) 실제값(y, L)이 0이고, 예측값(H(x), S)이 1인 경우
Logistic Cost -> -(1-0)log(1-1) = -log(0) = infinite
Cross Entropy -> Sum [1 // 0] * [-log(0) // -log(1)] = Sum [infinite // 0] = infinite

3) 실제값(y, L)이 1이고, 예측값(H(x), S)이 0인 경우
Logistic Cost -> -(1)log(0) = -log(0) = infinite
Cross Entropy -> Sum [0 // 1] * [-log(1) // -log(0)] = Sum [0 // infinite] = infinite

4) 실제값(y, L)이 1이고, 예측값(H(x), S)이 1인 경우
Logistic Cost -> -(1)log(1) = -log(1) = 0

Cross – entropy

One Hot Encoding

color
red
green
blue
red

one-hot → encoding

color_red	color_green	color_blue
1	0	0
0	1	0
0	0	1
1	0	0

Input Data size를 벡터 차원으로 하고, 나타낼 Class Index에 1을 부여하고, Others는 0을 부여

```
a = hypothesis ( x_data )

print ( a )
print ( tf.argmax( a, 1 ) )
print ( tf.argmax( y_data, 1) ) # same as print ( tf.argmax( a, 1 ) )
```

```
tf.matmul( X, W ) + b
hypothesis = tf.nn.softmax( tf.matmul ( X, W ) + b  ) = y = logits)


   # Cross – entropy cost/loss
1. cost = tf.reduce_mean( -tf.reduce_sum( Y  * tf.log(hypothesis), axis = 1 ) )

2. cost_i = tf.nn.softmax_cross_entropy_with_logits_v2( logits = logits,
                                                        labels = y_one_hot )
   cost = tf.reduce_mean( cost_i )


   # y_one_hot = tf.one_hot( list( y_data ), nb_classes ) 'error'
     y_one_hot = tf.reshape( y_one_hot, [ -1, nb_classes ]
```

Shape 때문에 One hot encoding error 발생 방지

```
import tensorflow as tf

data = [1, 2, 0, 2, 1]

num_classes = len(set(data)) # = 3, data 0~2 class 개수

one_hot_encoded = tf.one_hot(data, depth=num_classes)
```

Implementation

```
# tf.nn.softmax computes softmax activations
# softmax = exp(logits) / reduce_sum(exp(logits), dim)
def logit_fn(X):
    return tf.matmul(X, W) + b

def hypothesis(X):
    return tf.nn.softmax(logit_fn(X))

def cost_fn(X, Y):
    logits = logit_fn(X)
    cost_i = tf.keras.losses.categorical_crossentropy(y_true=Y, y_pred=logits,
                                                       from_logits=True)
    cost = tf.reduce_mean(cost_i)
    return cost
```

```
def grad_fn(X, Y):
    with tf.GradientTape() as tape:
        loss = cost_fn(X, Y)
        grads = tape.gradient(loss, variables)
        return grads

def prediction(X, Y):
    pred = tf.argmax(hypothesis(X), 1)
    correct_prediction = tf.equal(pred, tf.argmax(Y, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
    # This part is why use 'argmax', loss and accuracy

    return accuracy
```

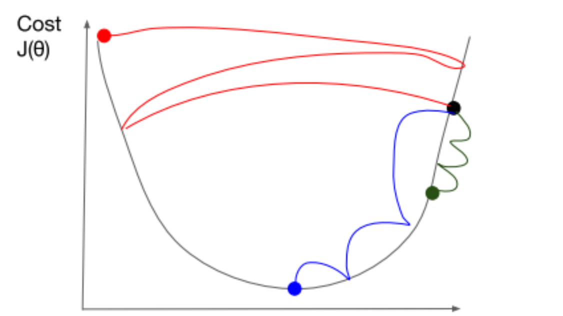
```
def fit(X, Y, epochs=1000, verbose=100):
    optimizer =  tf.keras.optimizers.SGD(learning_rate=0.1)

    for i in range(epochs):
        grads = grad_fn(X, Y)
        optimizer.apply_gradients(zip(grads, variables))
        if (i==0) | ((i+1)%verbose==0):
#              print('Loss at epoch %d: %f' %(i+1, cost_fn(X, Y).numpy()))
            acc = prediction(X, Y).numpy()
            loss = cost_fn(X, Y).numpy()
            print('Steps: {} Loss: {}, Acc: {}'.format(i+1, loss, acc))

fit(x_data, Y_one_hot)
```

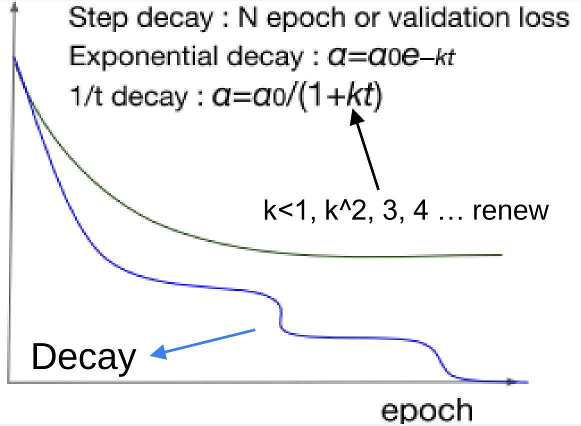
Learning Rate

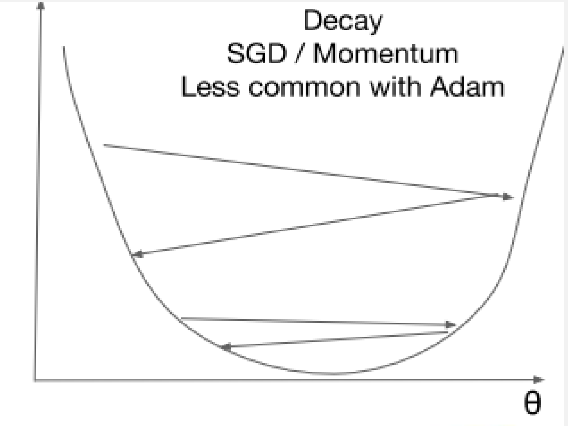
is a hyper-parameter that controls how much we are adjusting the weights with respect the loss gradient

Good vs Bad


Cost
J(θ)
High
Low
Good


Cost
J(θ)


Step decay : N epoch or validation loss
Exponential decay : $\alpha=\alpha_0 e^{-kt}$
1/t decay : $\alpha=\alpha_0/(1+kt)$
k<1, k^2, 3, 4 … renew
Decay
epoch

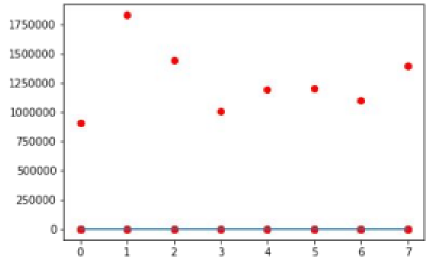

Decay
SGD / Momentum
Less common with Adam
θ

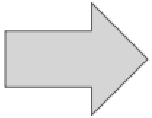
Data preprocessing

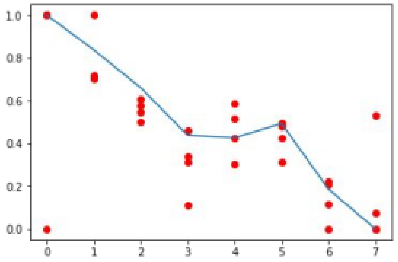
데이터 전처리

Google Developer Student Clubs

Feature Scaling







Standardization
(Mean Distance)

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

Normalization
(0~1)

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

[Python Code(numpy)]

```
Standardization = (data - np.mean(data)) / sqrt(np.sum(
(data - np.mean(data))^2 ) / np.count(data))
```

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

Overfitting

Overfitting

- Get more training data
- Smaller set of features
- Add additional features

Overfitting Solution

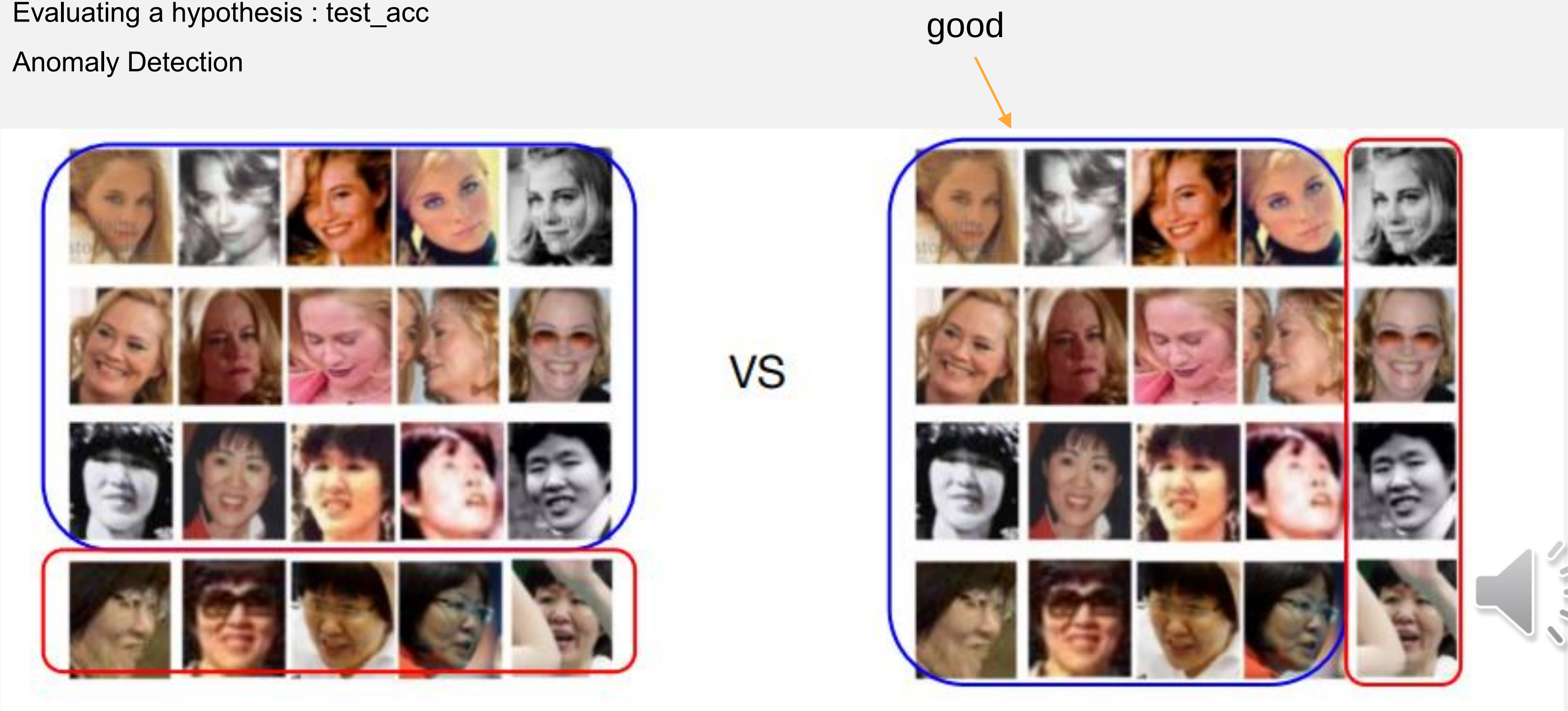
- Feature Normalization
- Regularization
- More Data and Data Augmentation – Lecture 9
- Dropout
- Batch Normalization

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

```
dataset = tf.data.Dataset.form_tensor_slices( ( x_train, y_train ) ).batch( len( x_train ) )
# .batch( len( x_train) ) : 데이터를 크기가 len(~)인 batch로 나눔

def l2_loss( loss, beta = 0.01 ) :
    W_reg = tf.nn.l2_loss( W )
    loss = tf.reduce_mean( loss + W_reg * beta )
    return loss
```

Data Set

- Training / Validation / Testing
- Evaluating a hypothesis : test_acc
- Anomaly Detection



Data Set : Anomaly Detection

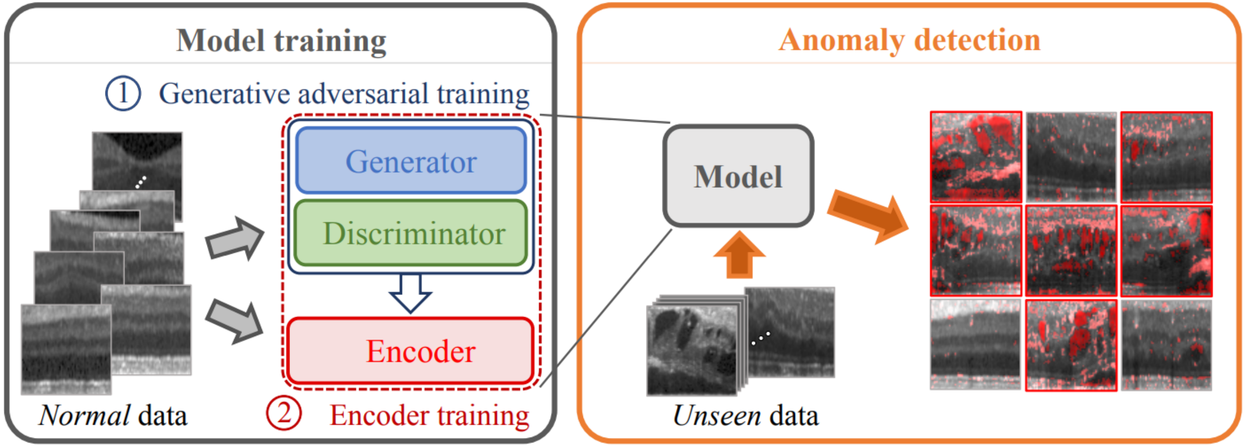


Figure 1: Anomaly detection framework. Both steps of model training, generative adversarial training (yields a trained generator and discriminator) and encoder training (yields a trained encoder), are performed on *normal* ("healthy") data and anomaly detection is performed on both, unseen healthy cases and anomalous data. (Best viewed in color)

Learning & Sample Implementation

Learning & Sample Data

Learning

- Online Learning vs Batch Learning
- Fine tuning : Feature Extraction
- Efficient models → 경량 model

Fine tuning :
- 기존에 학습된 ML model을 가져와서 새롭게 작업하거나 데이터에 맞게 조정하는 과정
- 기존 모델의 W, architecture는 유지

Sample Data

- Fashion MNIST, IMDB, CIFAR-100

.ipynb 과제

Review

- L2_loss ≈ MSE(Mean Squared Error)
- **One_hot part의 shape error**

1. 데이터 형태의 불일치 : 데이터가 1차원 배열 혹 list 형태로 주어짐
2. 출력 차원 불일치 : 모델 출력 layer에서 예상한 출력 차원
   = shape와 실제 one_hot_encoding 차원이 불일치
3. Input 데이터 형식 문제
4. 카테고리 값 개수 확인 : data의 class 수 확인