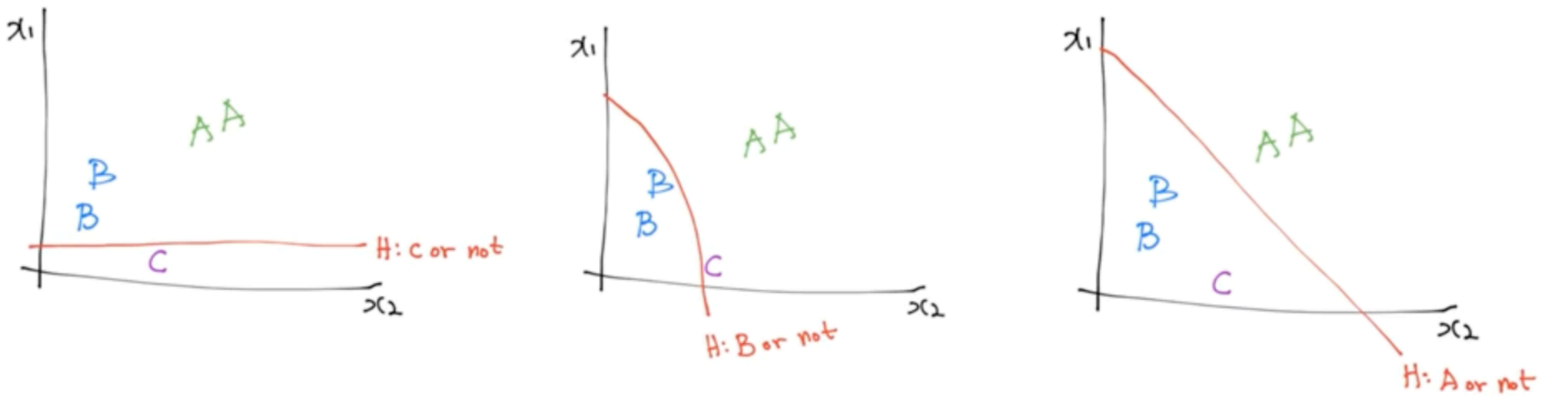
Week2 Presentation

Multinomial Classification
Learning Rate, Preprocessing

GDSC Hanyang ML/DL : Basic, Jaeseung Lee

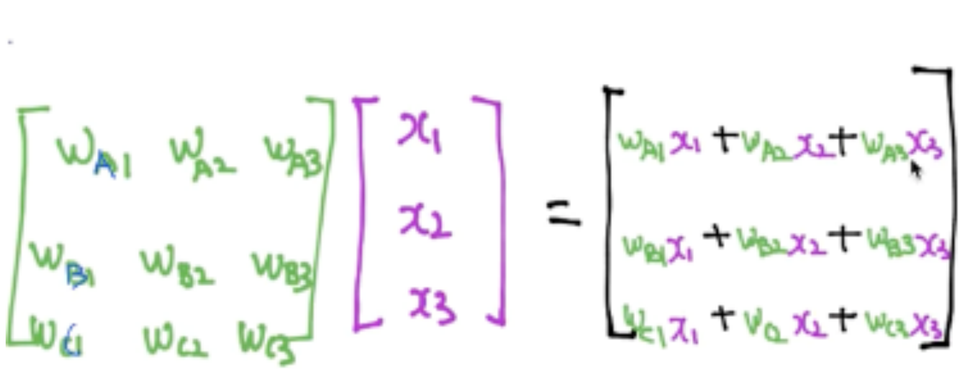
Index

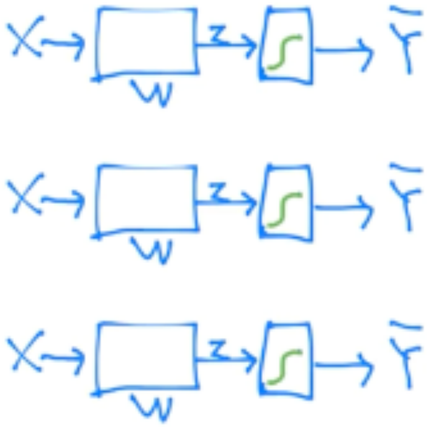
Multinomial Classification



- 여러 Label을 가지는 Classification을 Multinomial classification이라고 함.
- 이는 여러번의 Binary Classification으로 정의 가능함.
- 예제) A or not, B or not, C or not 이 세 번의 Binary Classification으로 Multinomial 구현

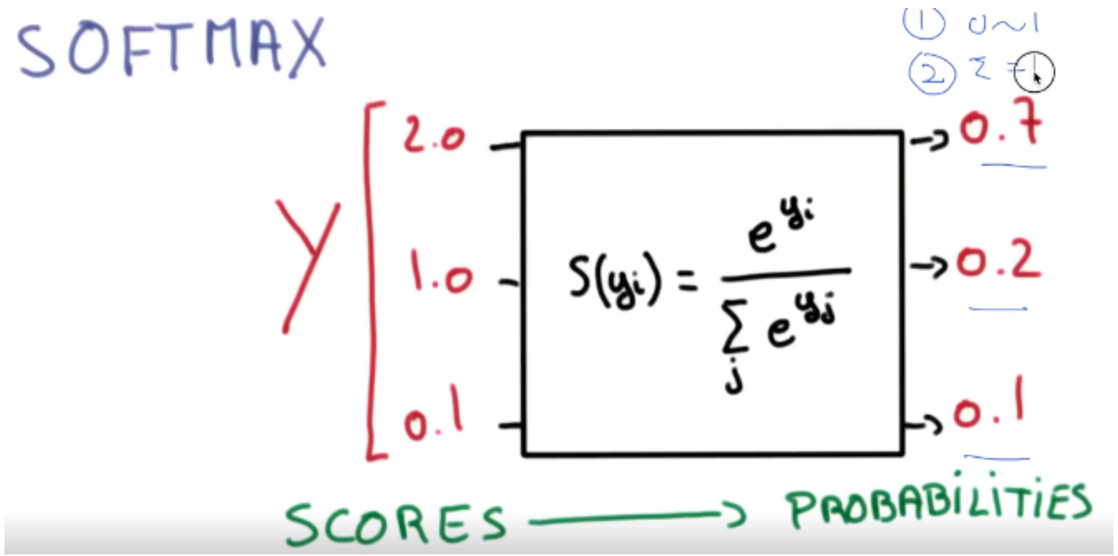
Multinomial Classification





- 원래는 w1, w2, w3 3개의 벡터를 사용해서 3번 연산했음.
- Weight의 row를 늘려 한번에 연산 가능하게 변환함.

Softmax

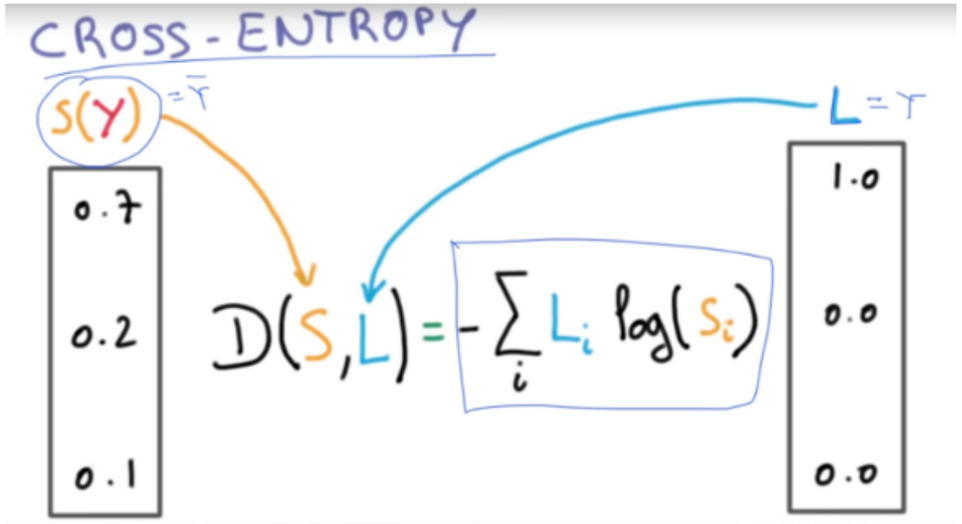


- Softmax란, 어떤 결과 값에 대응하여 그 probability를 구해주는 함수.
- 각 값들은 모두 0 ~ 1 사이의 값에 해당하며, 그 값들을 다 더했을 때 1이 나옴.
- 따라서 이를 확률, 비율로 사용 가능함.

Softmax

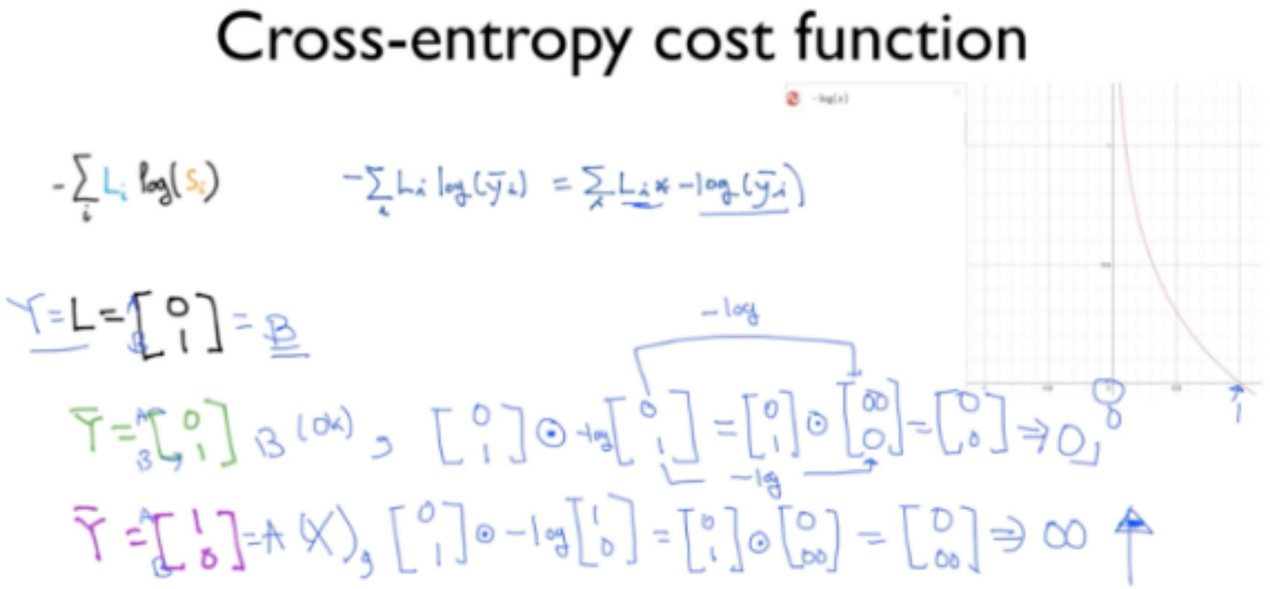
- 왜 Softmax를 사용해야 하는가.
  - Softmax는 Multinomial Classification에서 가장 큰 확률을 가지는 Feature를 쉽게 찾을 수 있게 해준다.
  - Argmax라는 함수(어떤 행렬의 최대값을 구함)와 같이 유용하게 사용할 수 있음.

Cross-entropy



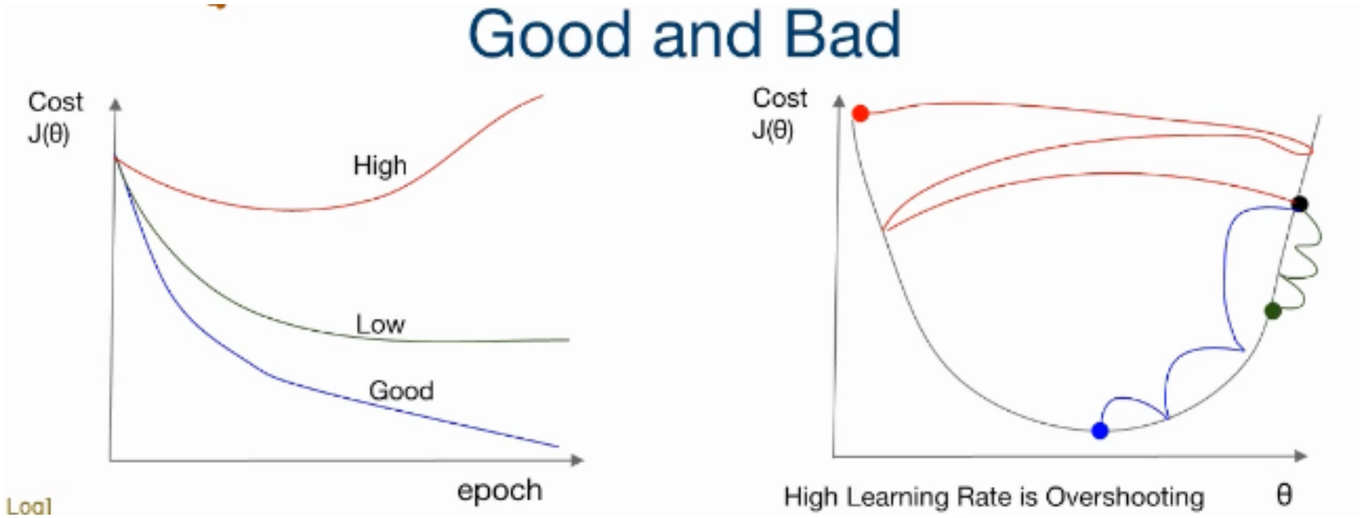
- Cross-entropy는 Cost function의 일종으로, 주로 Softmax와 함께 사용된다.
- S는 Softmax의 결과값, L은 실제 정답 Label을 뜻한다.

Cross-entropy



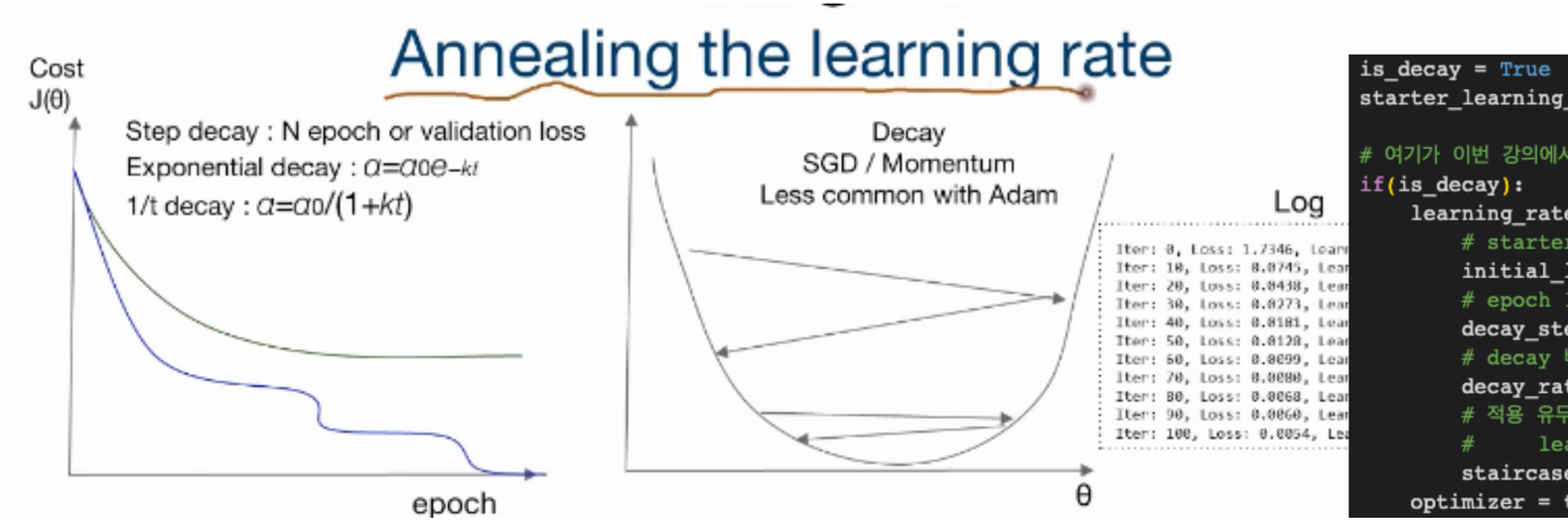
- 왜 log를 취하는가 : Softmax를 거치므로 범위는 0 ~ 1, 0 일때 무한대, 1일 때 0인 비용을 할당
- Element-wise multiplication (요소별 곱) : 행렬의 같은 row 끼리 곱셈을 하는 방식

Learning Rate



- Learning rate는 Gradient descent를 얼마나 적용할 지 결정하는 변수
- 너무 크면 발산하여 minimum에 도달 불가하고, 작으면 시간이 매우 오래 걸림

Learning Rate

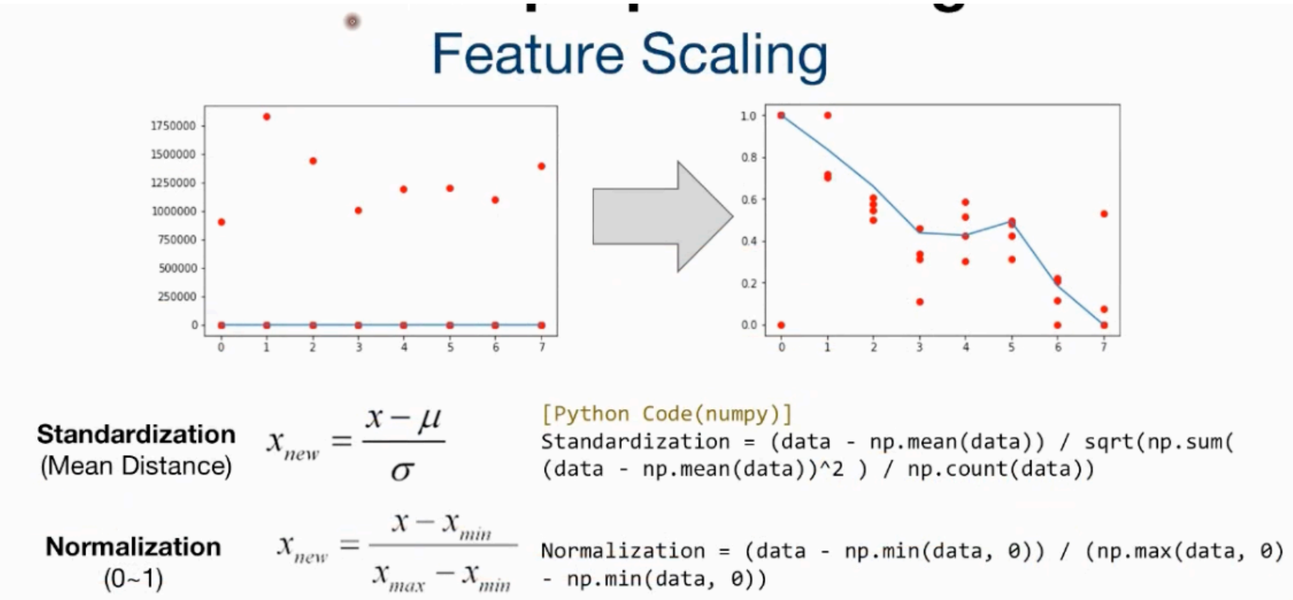



```python
is_decay = True
starter_learning_rate = 0.1

# 여기가 이번 강의에서 중요한 부분
if(is_decay):
    learning_rate = tf.keras.optimizers.schedules.ExponentialDecay(
        # starter_learning_rate : 0.1로 설정
        initial_learning_rate=starter_learning_rate,
        # epoch 1000번 마다 decay 해줌
        decay_steps=1000,
        # decay 비율은 0.96
        decay_rate=0.96,
        # 적용 유무 : decayed_learning rate =
        #     learning_rate * decay_rate ^ (global_step / decay_steps)
        staircase=True)
    optimizer = tf.keras.optimizers.SGD(learning_rate)
```

- 따라서 Learning rate를 상황에 따라 적절히 조절하는 것이 학습에 매우 유용하다.
- 이를 Learning rate decay라고 부르며, Step decay, Exponential decay 등이 있다.

Data Preprocessing




```python
def normalization(data):
    # 수식 : x = x-xmin / xmax - xmin
    numerator = data - np.min(data, 0)
    denominator = np.max(data, 0) - np.min(data, 0)
    return numerator / denominator
```

- 튀는 데이터들을 처리해주어야 학습에 있어서 효율적이고 정확한 학습이 된다.
- 그 방법으로는 Standardization(표준화)와 Normalization(정규화) 가 있다.

Overfitting

- Overfitting이란 모델이 데이터 셋에 너무 맞춰서 학습된 경우를 말한다.
- 이 경우에 dataset에 있는 data로 test시 결과는 매우 좋지만, dataset에 있지 않는 data로 test시 결과가 좋지 않다.

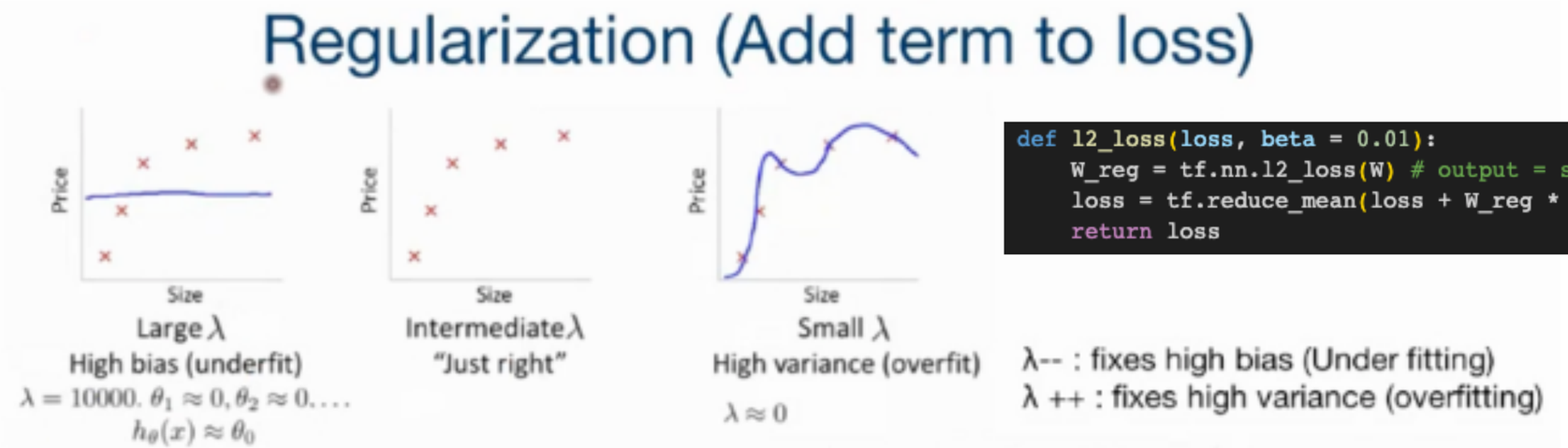
Overfitting

How to solve

- Get more train data
  - 결국에 train data가 많아질 수록, 더 많은 데이터들을 다루기 때문에 Overfitting 방지
- Smaller set of features
  - feature의 개수가 작을 수록, weigth의 개수도 줄어 variance가 줄어들게 된다.
  - PCA (주성분 분석) : 차원을 낮춰줌.

Overfitting

How to solve




```python
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) # l2_loss weight에 적용
    return loss
```

- Regularization 방식은 Loss function에 어떤 term을 두어 튀는 변수들을 제어함.
- 이는 Loss function의 안정화에 기여함.

Data sets & Learning

Data sets

- Training / Validation Data set partitioning
  - 어떠한 dataset에서 제대로 Train과 Validation set을 나누는 것이 중요하다.
- Anomaly detection (이상 감지)
  - GAN Network를 통해 Unusual data를 탐지하여 Dataset을 깔끔하게 한다.
  - 적대적 생성 신경망이라고 하며, 두 시스템의 경쟁을 통해 학습한다.

Data sets & Learning

Learning

	Online Learning	Batch(Offline) Learning
Data	Fresh	Static
Network	connected	disconnected
Model	Updating	Static
Weight	Tunning	initialize
Infra(GPU)	Always	Per call
Application	Realtime Process	Stopping
Priority	Speed	Correctness

- Online vs Batch Learning
- Online은 실시간으로 업데이트 되며, 속도를 중요시할 때 사용.
- Batch는 Offline에서 Static하게 Update하며, 정확성을 중요시 할 때 사용.

Data sets & Learning

Learning

- Fine Tuning / Feature Extraction
  - Fine tuning은 기존의 weight 값을 미세 조정하는 방식을 뜻함. 기존의 모델에 새로운 것들을 추가 할 때 기존 weight만 조금 바꿔 다른 데이터 셋들도 처리 가능하게 함.
  - Feature Extraction은 특징을 추출하는 방식으로, 특정 특징에 대해서만 추가로 학습하게 하는 방식
- Efficient Models
  - 실제 Field에서는 Model의 경량화가 서비스의 속도를 좌지우지 하기 때문에, fully connected layer에서 1x1 convolution으로 결과를 도출하면 size 줄어들어 경량화!

Thank you for Listening

Multinomial Classification
Learning Rate, Preprocessing

GDSC Hanyang ML/DL : Basic, Jaeseung Lee