Deep Learning Tutorial #5

Ref.

- Collado, Julian, et al. "Learning to identify electrons." Physical Review D 103.11 (2021): 116028.
- Collado, Julian, et al. "Learning to isolate muons." Journal of High Energy Physics 2021.10 (2021): 1-17.

IAM

- 한상곤(Sangkon Han, sangkon@pusan.ac.kr), CS.

Day.5 (practice) `Learning to Isolate Muons`

> `Learning to Isolate Muons` 논문을 재현해 보자.

Index

- ~~Day.1 : (Intro) - Hands-On~~
- ~~Day.2 : (Example) - `LeNet-5` 구현해보기~~
- ~~Day.3 : (execrcise) - `Learning to Identify Electrons` 재현 준비~~
- ~~Day.4 : (execrcise) - `Learning to Identify Electrons` 재현~~
- Day.5 : (practice) - `Learning to Isolate Muons` 재현

`Learning to Isolate Muons` 재현 연습

Step1. 만들고자 하는 구조를 미리 정의하라.

- 데이터
  - (https://mlphysics.ics.uci.edu/data/2021_muon/)
- 훈련
  - PFN(What is this?)
- 평가
  - ROC/AUC

Step.2.a

- H5 파일은 충분히 해결할 수 있음

```
with h5py.File("data.h5", "r") as hf:
    for key in list(hf.keys()):
        for dset in ['train', 'valid', 'test']:
            data = hf_save[key+'/'+dset][:]
['images_bg', 'images_signal', 'iso_bg', 'iso_signal', 'weights_bg', 'weights_signal']
```

Step.2.b `npy`

`npy` 파일은 `NumPy` 라이브러리가 설치된 Python 소프트웨어 패키지로 작성된 NumPy 배열로 만들어진 파일

```
import numpy as np

x_data = np.arange(5)
y_data = np.sin(x)

np.savez('xy.npz', x=x_data, y=y_data)
data = np.load('xy.npz')

x, y = data['x'], data['y']
```

Step.2.a

- `PFN`의 경우 `EnergyFlow`에서 제공하는 `Particle Flow Network`를 뜻 함
- `PFN`은 어떻게 알 수 있나요?

Step.2.b

- GitHub에서 제공된 코드를 확인.
- `PFN`, `EFN`의 경우 해당 라이브러리에서 제공하는 문서를 확인

Step.2.c

```
class MuonImageNetwork(Sequential):
    def __init__(self):
        super().__init__()
        self.add(Conv2D(48, 3, input_shape=(32,32,1), activation = 'relu'))
        self.add(Conv2D(48, 3, input_shape=(32,32,1), activation = 'relu'))
        self.add(Conv2D(48, 3, input_shape=(32,32,1), activation = 'relu'))
        self.add(MaxPooling2D(2,strides=2))
        self.add(Dense(74,activation="relu"))
        self.add(Dense(74,activation="relu"))
        self.add(Dense(1, activation="sigmoid"))
        self.compile(optimizer='adam', loss="binary_crossentropy", metrics=[tf.keras.metrics.AUC()])
```

Step.2.d

```
model=MuonImageNetwork()
model.summary()
```

Step.2.e

```
Model: "muon_image_network"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 48)        480

 conv2d_1 (Conv2D)           (None, 28, 28, 48)        20784

 conv2d_2 (Conv2D)           (None, 26, 26, 48)        20784

 max_pooling2d (MaxPooling2D  (None, 13, 13, 48)       0
 )

 dense (Dense)               (None, 13, 13, 74)        3626

 dense_1 (Dense)             (None, 13, 13, 74)        5550

 dense_2 (Dense)             (None, 13, 13, 1)         75

=================================================================
Total params: 51,299
Trainable params: 51,299
Non-trainable params: 0
_________________________________________________________________
```

Step.2.f

```
es = tf.keras.callbacks.EarlyStopping(
    monitor="val_loss", mode="auto", verbose=0, patience=5
)

model.fit(
    X,
    y,
    batch_size=256,
    epochs=100,
    verbose=0,
    validation_split=0.25,
    callbacks=[es],
)
```

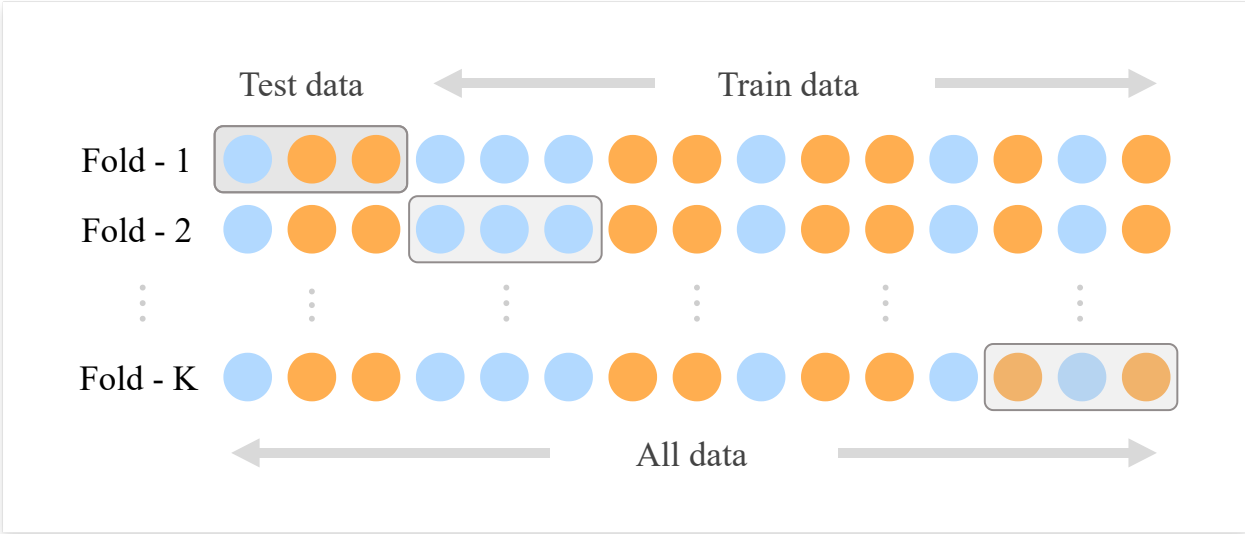
Step.3

```
predictions = np.hstack(model.predict(test_X))
```

측정에 관하여(AUC, ROC)

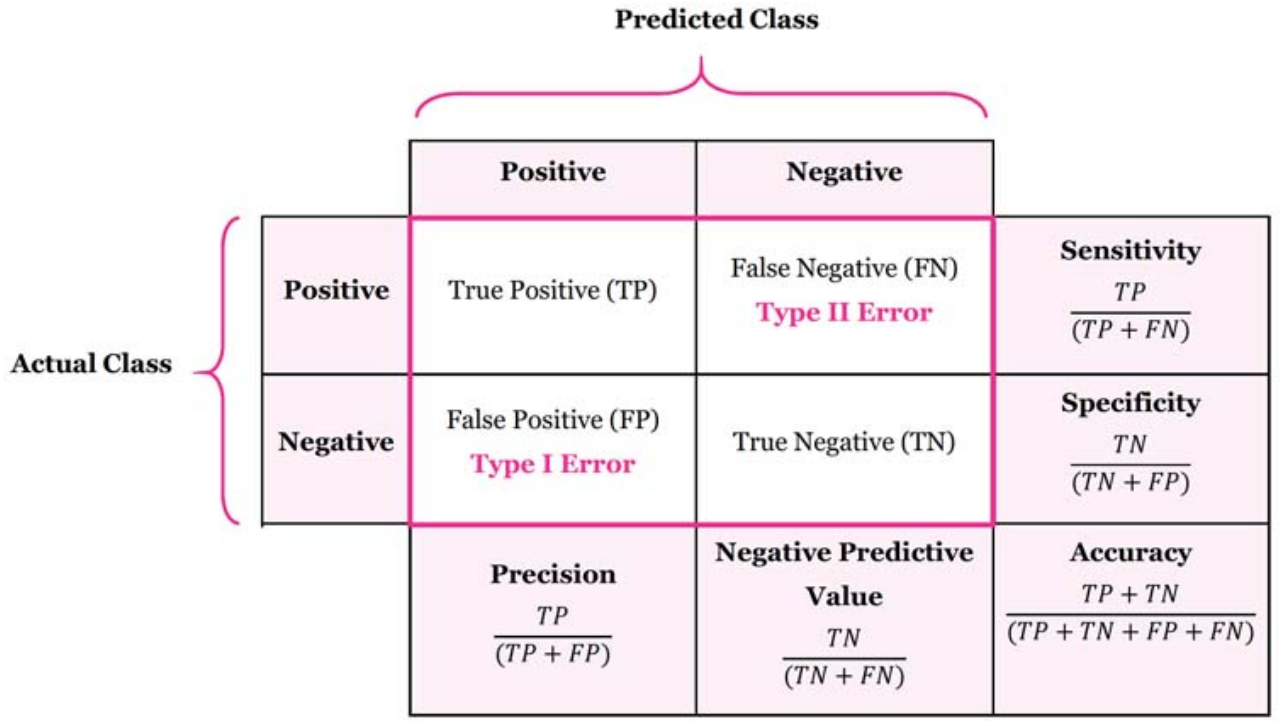
K-Fold Cross Validation

- K개의 폴드 세트에 K번의 학습과 검증 평가 반복 수행
- `Stratified K-Fold` 불균형한 분포도를 가진 레이블 데이터 집합을 위한 K-Fold 방식, 학습데이터와 검증 데이터를 나눌 때 각 레이블의 비율을 맞춰 나눔



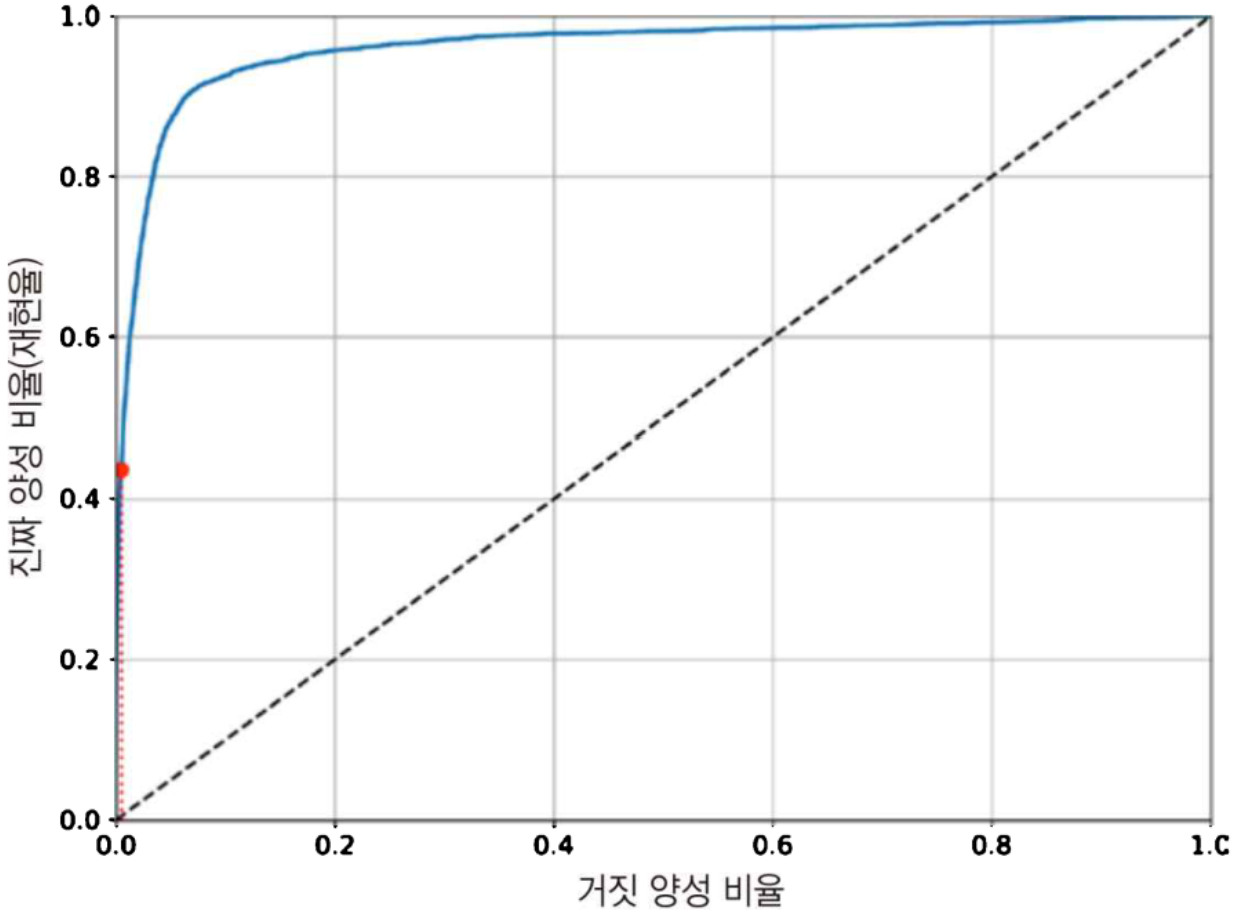
Confusion Matrix

- `Accuracy` (정확도), 전체 모델 중 바르게 분류한 비율
- `Precision` (정밀도), Positive라 분류한 것 중 실제값이 Positive인 비율
- `Recall` (재현도), 실제값이 Positive인 것 중 모델이 Positive라 분류한 비율
- `F1` , Precision과 Recall의 조화평균 (단 F1의 경우 데이터 불균형일 때 사용)



ROC(수신기 조작 특성)

- ROC 곡선은 정밀도에 대한 재현율 곡선이 아니고, 거짓 양성 비율(FRP)에 대한 진짜 양성 비율(TPR)의 곡선
  - FRP, 양성으로 잘못 분류된 음성 샘플의 비율
  - `1` 에서 음성으로 정확하게 분류한 음성 샘플의 비율인 진짜 음성 비율(TNR)을 뺀 값, 그래서 TNR을 특이도(specificity)라고 함
- 그러므로 ROC 곡선은 민감도(재현율)에 대한 1-특이도 그래프



AUC(area under the curve)

- 곡선 아래의 면적을 측정하면 분류기들을 비교할 수 있음
  - 완벽한 분류기는 ROC의 AUC가 1이고, 완전한 랜덤 분류기는 0.5임

Caution

ROC 곡선이 정밀도/재현율(PR) 곡선과 비슷해서 어떤 것을 사용해야 할지 궁금할 수 있는데, 일반적인 법칙은 양성 클래스가 드물거나 거짓 음성보다 거짓 양성이 더 중요할 때 PR 곡선을 사용하고 그렇지 않으면 ROC 곡선을 사용해야 함

하이퍼 파라미터(HyperParameter)

- 하이퍼 파라미터는 모델링할 때 사용자가 직접 세팅해주는 값
- 하이퍼 파라미터는 정해진 최적의 값이 없고, 휴리스틱한 방법이나 경험 법칙(rules of thumb)에 의해 결정하는 경우
- Keras Tuner와 같은 간단한 도구는 알아두는 것이 좋음

하이퍼 파라미터(HyperParameter) 종류

- 학습률(Learning Rate)
  - gradient의 방향으로 얼마나 빠르게 이동할 것인지 결정하는 변수, 너무 작으면 학습의 속도가 늦고, 너무 크면 학습 불가
- 손실 함수(Loss/Cost Function)
  - 입력에 따른 기대 값과 실제 값의 차이를 계산하는 함수
- 미니 배치 크기(Mini-batch Size)
  - 배치셋 수행을 위해 전체 학습 데이터를 등분하는(나누는) 크기, 가용 메모리 크기와 epoch 수행 성능을 고려
- 훈련 반복 횟수(Training Loop)
  - 학습의 조기 종료를 결정하는 변수, 학습 효율이 떨어지는 시점을 적절히 판단
- 은닉층의 뉴런 개수(Hidden Unit)
  - 훈련 데이터에 대한 학습 최적화 결정 변수, 첫 Hidden Layer의 뉴런 수가 Input Layer 보다 큰 것이 효과적

하이퍼 파라미터(HyperParameter) 종류

- Manual Search
  - 휴리스틱 조합, 사용자의 직관과 경험 기반 탐색
  - 탐색의 단순성 적용, 사용자 도출 조합 중 최적 조합 적용
- Grid Search
  - 모든 조합 탐색, 하이퍼파라미터 적용값 전체 탐색
  - 시행 횟수 한계 파악, 하이퍼파라미터 증가로 인해 전수 탐색 한계
- Random Search
  - 랜덤 샘플링, 범위 내 무작위 값 반복 추출
  - 탐색 범위 부여, 하이퍼파라미터 최소/최대값부여
- Bayesian Optimization
  - 관측 데이터 기반 F(x) 추정, 베이즈 정리 확용/가우시안 프로세스
  - 함수 생성, 확률 추정 결과 기반 입력값 후보 추천 함수

Conclusion

- 데이터 처리에 각별히 주의를 기울이세요.
  - 배포(deploy)를 생각하세요.
  - 문자열 보다는 이진파일을 고려하세요.
- `Python` 보다는 `CNN` 에 집중하세요.
  - 데이터, 훈련, 측정 3단계로 구성하세요.
  - 모든 단계를 빠르게 연결하세요.(Input - Conv2D - Pooling - Flatten - Dense)
  - 작동하는 작은 모델로 시작하고, 하이퍼 파리머터를 고려하세요.
- 측정 방식에 유의하세요.
  - 데이터에 따라 측정 방식을 결정하세요.
  - `plt.savefig('destination_path.{eps, jpg, pdf}', format='{eps, jpg, pdf}')`