[BAT512] Advanced Data Mining with AI

11강

Techniques for performance improvement

UNIST 융합경영대학원 2023학년도 2학기

UNIST 융합경영대학원
이규민(Gyumin Lee)
glee.optimizt@gmail.com

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

목차

1. 하이퍼파라미터 튜닝
2. K-fold Cross validation
3. Early stopping
4. Dropout
6. (실습) PyTorch

1. 하이퍼파라미터 튜닝

개요

❖ 정의

- 머신러닝 모델에 대한 최적의 하이퍼파라미터 조합을 탐색하고 선정하는 과정
- 하이퍼파라미터 최적화(optimization) 라고도 함

❖ 하이퍼파라미터 vs. 파라미터

	하이퍼파라미터 (Hyperparameter)	파라미터 (Parameter)
설명	- 초매개변수 - 머신러닝 모델링을 위해 사용자가 직접 선택하는 값 - 머신러닝 모델 학습 전 선정	- 매개변수 - 머신러닝 모델 내부의 변수 데이터로부터 결정 - 머신러닝 모델 학습의 대상
예시	- 학습률 - 배치 크기 - 인공신경망 구조 등	- 정규분포의 평균, 표준편차 - 선형 회귀 계수 - 가중치, 편향
조정 가능 여부	O	X

1. 하이퍼파라미터 튜닝

개요

❖ 하이퍼파라미터의 종류

- 선형/로지스틱 회귀 모델
  - 정규화(Regularization) 여부 및 종류
    - L1 norm, L2 norm
  - 학습 방식(로지스틱 회귀)
    - 경사하강법 등
- 의사결정나무/랜덤 포레스트
  - 분할 기준
    - Gini 계수, Entropy 등
  - 최대 깊이
  - 분할 최소 샘플 수
  - 트리 수(랜덤 포레스트)
- 클러스터링
  - 클러스터 수(K-means clustering)
  - 군집의 반경(epsilon)
  - 반경 내 최소 샘플 수

1. 하이퍼파라미터 튜닝

개요

❖ 하이퍼파라미터의 종류

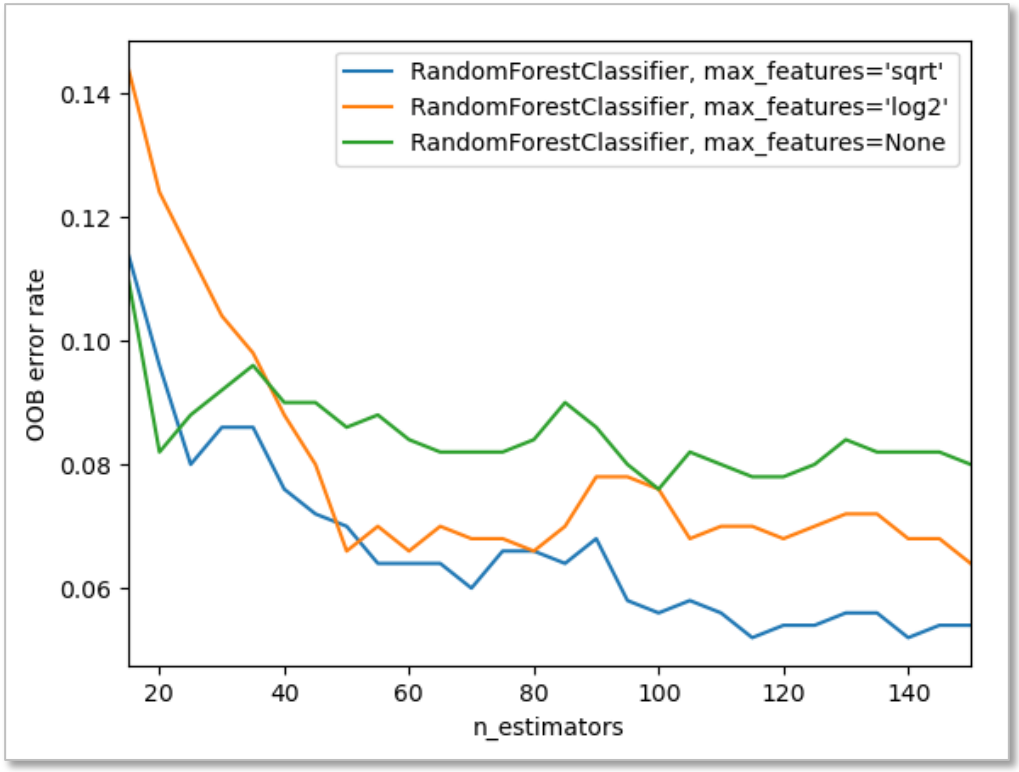
- 인공신경망(ANN, CNN, RNN, ...)
  - 배치 크기
  - 학습률(Learning rate)
  - 은닉층 수
  - 은닉 노드 수
  - (최대) 학습 횟수
  - 손실 함수
    - Mean squared error, Cross-entropy, Negative log loss 등
  - 학습 방식
    - Stochastic gradient descent, Momentum, Adagrad, Adam 등

1. 하이퍼파라미터 튜닝

필요성

❖ 모델 성능 개선

- 같은 머신러닝 모델이라도 어떤 하이퍼파라미터를 사용하는지에 따라 성능이 달라짐
  - 학습에 활용하는 데이터에 적합한 형태로 모델을 구축하고, 학습 방식을 정의해야함

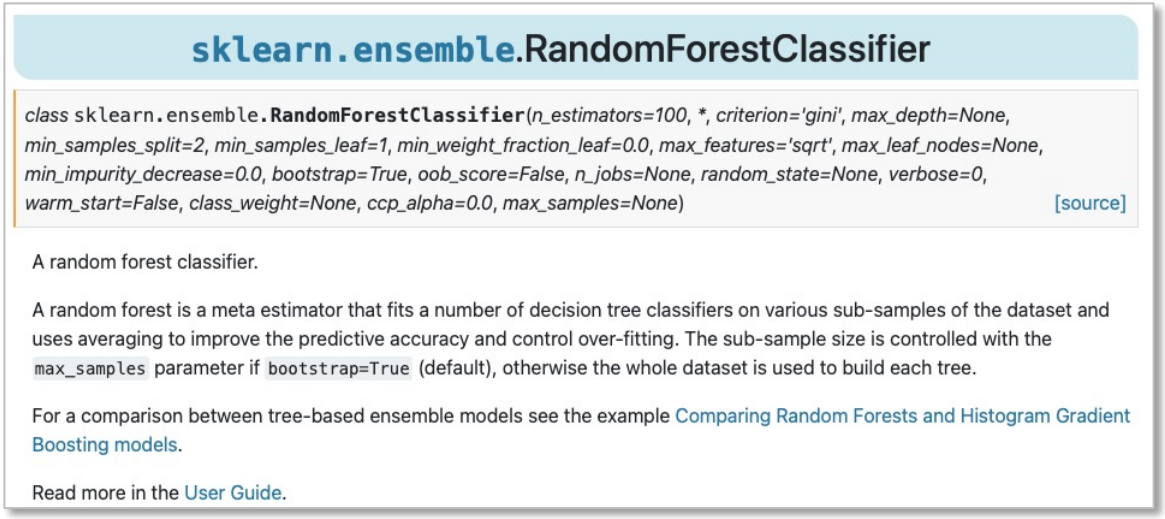


1. 하이퍼파라미터 튜닝

방법론

❖ Manual search (Rules of thumb)

- 경험에 따라 특정 형태의 데이터, 특정 머신러닝 모델에 적합한 하이퍼파라미터를 선정하는 방식
- 대부분의 경우 특정 머신러닝 모델이 좋은 성능을 내도록 하는 하이퍼파라미터 값들이 알려져 있음
  - scikit-learn과 같은 머신러닝 라이브러리에서는 이를 기본값으로 제공
- 간편하게 모델 학습 단계로 진입하고 실험 결과를 빠르게 알 수 있으나, 해당 결과가 최적의 결과임을 보장하지 않음
  - 파일럿 테스트 시 주로 활용



sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

For a comparison between tree-based ensemble models see the example Comparing Random Forests and Histogram Gradient Boosting models.

Read more in the User Guide.

1. 하이퍼파라미터 튜닝

방법론

❖ Grid search

- 머신러닝 모델 구축에 필요한 하이퍼파라미터의 종류와 값의 후보군을 지정하고, 가능한 모든 조합을 대상으로 학습 및 성능 평가를 실시하여, 가장 성능이 좋은 조합을 최종 하이퍼파라미터로 선정하는 방법
- 가능한 모든 조합을 탐색하므로 최적의 하이퍼파라미터 조합을 찾을 수 있으나, 하이퍼파라미터 값의 후보군을 적절히 설정하기 어려우며 탐색 시간이 매우 오래 걸림

```
# Tuning hyper-parameters for precision

Best parameters set found on development set:

{'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}

Grid scores on development set:

0.986 (+/-0.016) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.959 (+/-0.028) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.988 (+/-0.017) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.982 (+/-0.026) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
0.988 (+/-0.017) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.983 (+/-0.026) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
0.988 (+/-0.017) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.983 (+/-0.026) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
0.974 (+/-0.012) for {'C': 1, 'kernel': 'linear'}
0.974 (+/-0.012) for {'C': 10, 'kernel': 'linear'}
0.974 (+/-0.012) for {'C': 100, 'kernel': 'linear'}
0.974 (+/-0.012) for {'C': 1000, 'kernel': 'linear'}
```

1. 하이퍼파라미터 튜닝

방법론

❖ Random search

- Grid search와 유사하지만, 하이퍼파라미터 값의 후보군을 직접 지정하지 않고 값의 범위만을 지정하여, 해당 범위 내에서 임의의 값을 선택하여 탐색하는 방식
- 탐색 횟수를 지정하여 탐색을 수행하므로, grid search에 비해 빠른 속도로 원하는 수준의 성능을 보이는 하이퍼파라미터 조합을 찾을 수 있음
- 반복 횟수 내 최적 조합이 나타나지 않을 가능성이 존재함

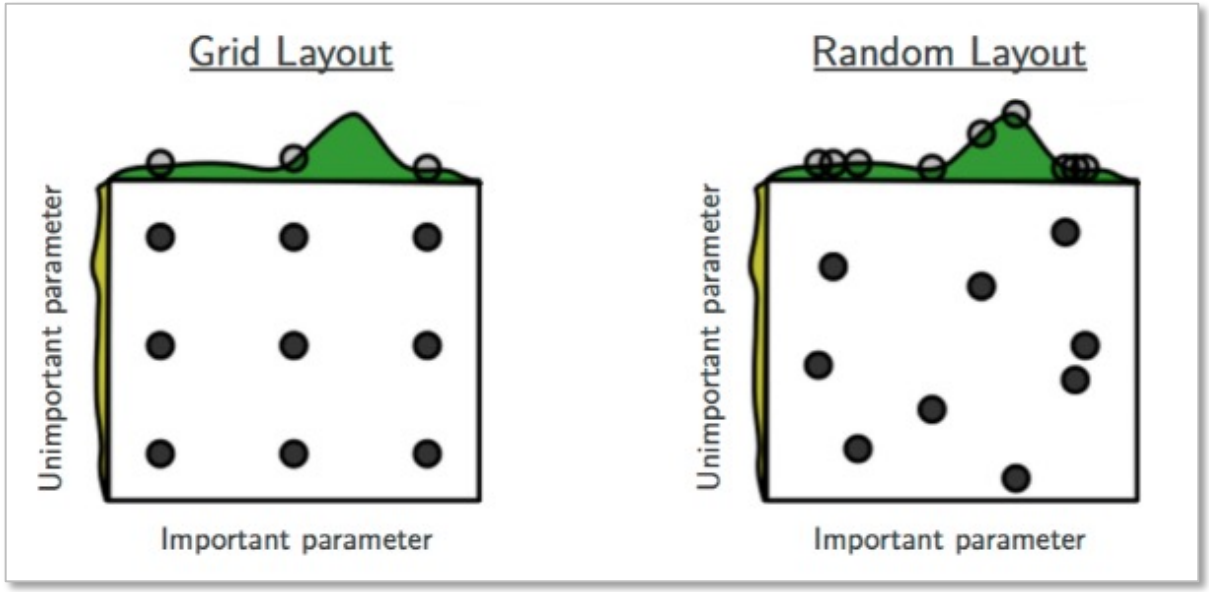
```
random_search =
RandomizedSearchCV(RandomForestRegressor(random_state=0),
                                {
                                   'n_estimators':np.arange(5,100,5),
                                   'max_features':np.arange(0.1,1.0,0.05),
                                 },cv=5, scoring="r2",verbose=1,n_jobs=-1,
                                  n_iter=50, random_state = 0
                                )
random_search.fit(X_train,y_train)
```

1. 하이퍼파라미터 튜닝

방법론

❖ Grid search vs. Random search

- 모델 성능에 대한 각 하이퍼파라미터의 중요도가 다를 수 있음
- Grid search는 모든 하이퍼파라미터에 대해 동일한 횟수로 탐색
  → 중요하지 않은 하이퍼파라미터에 대해서도 많은 시간을 할애
- Grid search를 통해 대략적인 가이드라인을 정하고, 이후 random search를 사용하거나, 출력 변수와의 상관관계가 높은 하이퍼파라미터에 대해 우선적으로 튜닝을 실시하기도 함

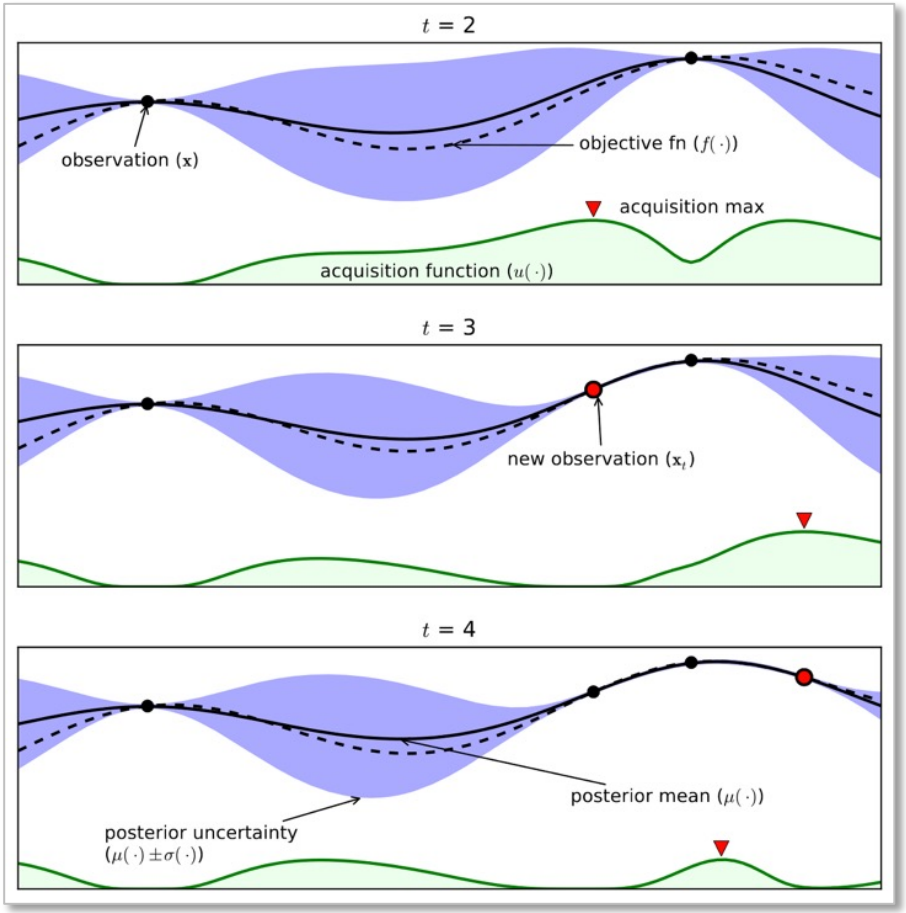


1. 하이퍼파라미터 튜닝

방법론

❖ Bayesian optimization

- Bayes rule을 바탕으로, 미지의 함수가 반환하는 값의 최대값을 탐색하기 위한 최적화 기법
  - 데이터 관측치를 하나씩 관찰하면서 사후 확률(posterior probability) 분포를 업데이트하여 전역해(global optimum)의 근사값을 얻음
- Grid/random search에 비해 큰 규모의 데이터셋에서 최적의 하이퍼파라미터 조합을 찾는 데 걸리는 시간을 대폭 줄일 수 있음
- 하지만 Bayesian optimization 을 수행하는 데에도 또 다른 하이퍼파라미터가 필요
  - Surrogate model
  - Acqusition function



2. K-fold Cross validation

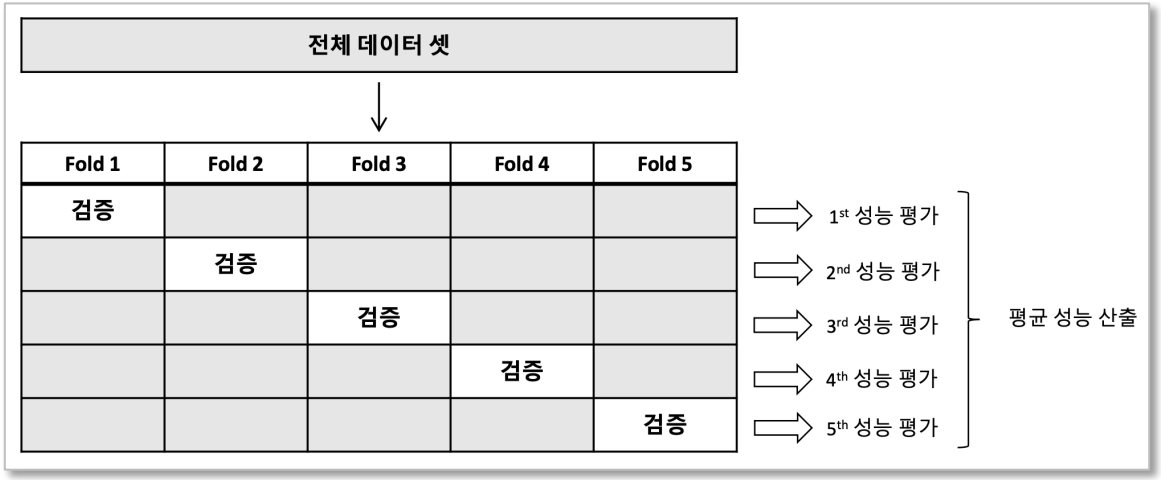
개요

❖ 정의

- 데이터셋을 K개 부분 집합으로 나누어 머신러닝 모델의 학습 및 검증을 여러 번 수행하는 방법
  - 데이터셋의 모든 데이터 샘플을 최소 한 번 이상 검증에 활용할 수 있음

❖ 필요성

- 데이터가 많지 않은 경우, 학습/검증/테스트 데이터셋 분할이 어려울 수 있음
  → K-fold cross validation을 통해 최대한 많은 데이터를 모델 학습 및 검증에 사용할 수 있음
- 서로 다른 데이터에 대한 모델의 성능 평가 결과를 취합하여 평가하므로, 모델의 일반화 성능을 평가하는데 유용함

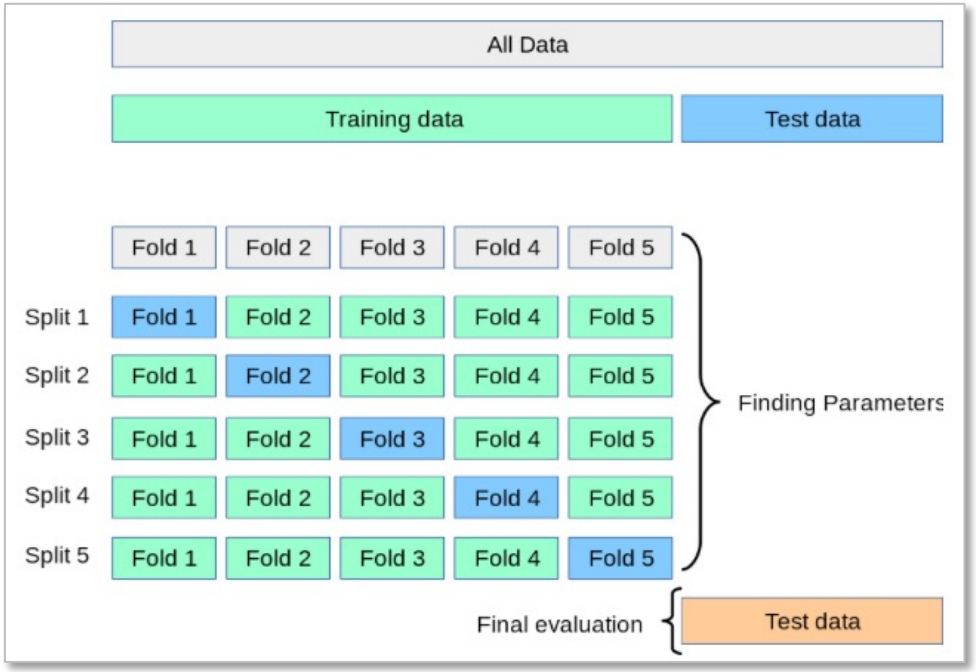


2. K-fold Cross validation

구현 방식

❖ K-fold cross validation

- 전체 데이터셋을 학습용/테스트용 데이터셋으로 분할
- 학습용 데이터셋에 K-fold cross validation을 수행하여 하이퍼파라미터 튜닝
- 최적의 하이퍼파라미터 조합으로 모델을 구축하고, 학습용 데이터셋 전체에 대해 최종 학습
- 테스트용 데이터셋을 활용하여 학습이 완료된 모델의 성능을 평가

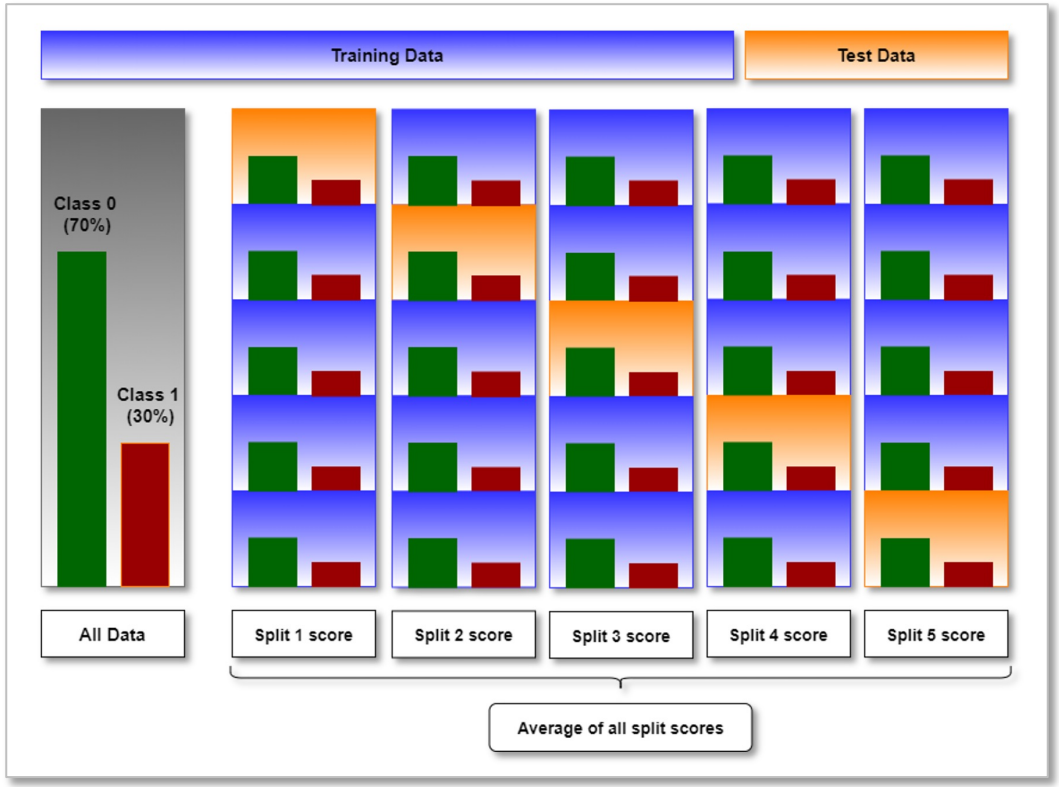


2. K-fold Cross validation

구현 방식

❖ Stratified K-fold cross validation

- K-fold cross validation과 같은 방식이지만, 데이터셋을 여러 부분 집합으로 나눌 때 학습 데이터와 검증 데이터에 포함된 데이터 샘플의 레이블(범주) 분포가 같도록 무작위 샘플링(Stratified random sampling)을 수행하는 방식

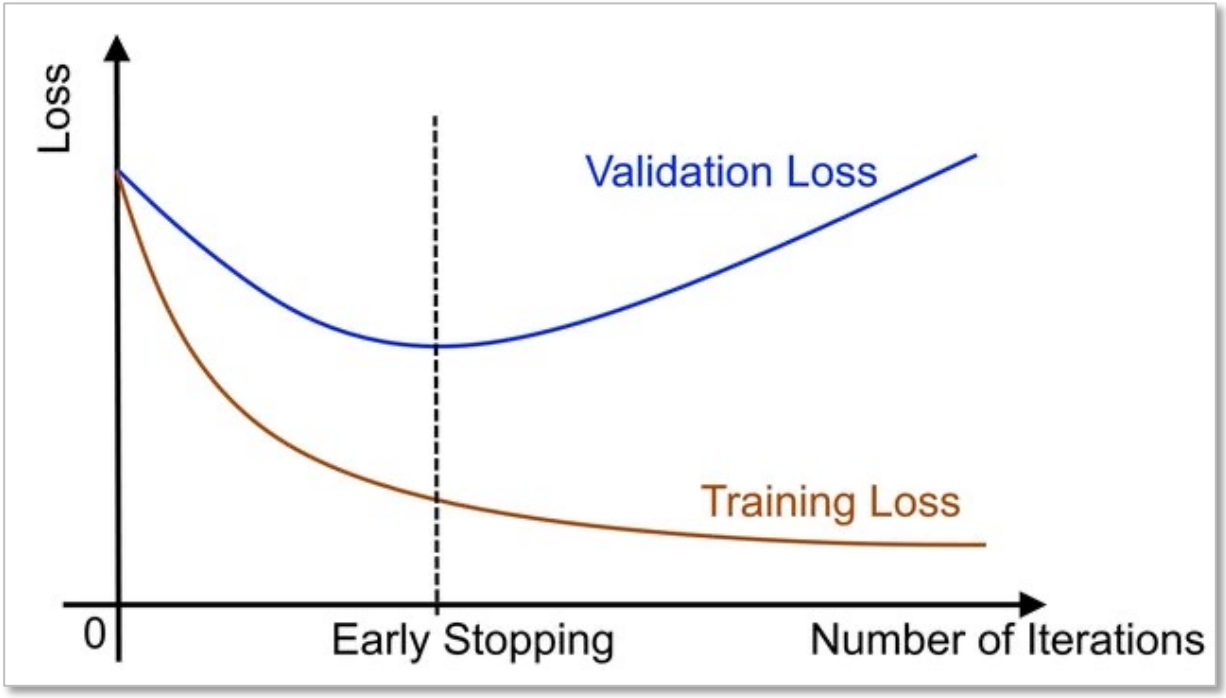


3. Early stopping

개요

❖ Training loss vs. Validation loss

- 일반적으로 머신러닝 모델의 학습이 진행됨에 따라, 학습 데이터셋에 대한 성능은 계속해서 높아지지만 검증 데이터셋에 대한 성능은 특정 시점 이후 더 이상 높아지지 않고 오히려 낮아짐
  → 학습 데이터셋에 과적합되기 시작



3. Early stopping

구현 방식

❖ 머신러닝 모델 학습 과정

- 모델 학습을 시작하기 전 조기 종료 조건(early stopping criterion)을 설정함
  - early stopping criterion: 특정 학습 횟수(epoch)를 지정하여, 해당 학습 횟수 동안 검증 데이터셋에 대한 성능이 개선되지 않으면 학습을 종료함
- 학습이 진행되는 동안 매 epoch마다 학습/검증 데이터셋에 대한 손실(loss)을 계산
- 검증 데이터셋에 대한 손실 값이 낮아지고 있는지 판단, early stopping patience를 업데이트
  - early stopping patience: 모델의 성능이 개선되지 않은 상태로 진행된 학습 횟수. 성능이 개선되었다고 판단되는 순간 초기화됨
  - min_delta: 모델의 성능이 개선되고 있다고 판단하기 위한 최소 변화량
- 이전 epoch에 비해 모델 성능이 개선되었다면, 현 시점의 모델(best model)을 따로 저장해둠
- early stopping patience가 early stopping criterion보다 커지는 순간 학습을 종료
- early stop이 발생하지 않았다면 최종 학습 시점의 모델을 사용하고, early stop이 발생하였다면 학습 종료 시점 모델 대신 이전 학습 epoch 중 가장 성능이 좋았던 모델을 불러와서 사용함

4. Dropout

개요

❖ 등장 배경

- 인공신경망의 은닉층/은닉노드의 수가 많아지면, 더욱 복잡한 문제를 해결할 수 있음
- 그러나 모델의 크기가 커질수록(깊고 복잡해질수록) 학습 데이터에 과적합될 가능성이 높고, 학습 시간도 길어지고, 필요한 데이터의 양도 많아짐 → 학습 효율 감소
- 모델의 성능은 최대한 유지하면서, 모델이 커지면서 생기는 단점을 보완하기 위한 방법으로 고안됨

❖ 정의

- 인공신경망에 대한 학습을 진행할 때, 모델에 존재하는 모든 뉴런(노드)에 대해 학습을 수행하지 않고, 일부를 생략(dropout)하여 보다 단순해진 모델에 대해 학습을 진행함
  → 의사결정나무/랜덤 포레스트 기법의 가지치기(pruning)과 유사

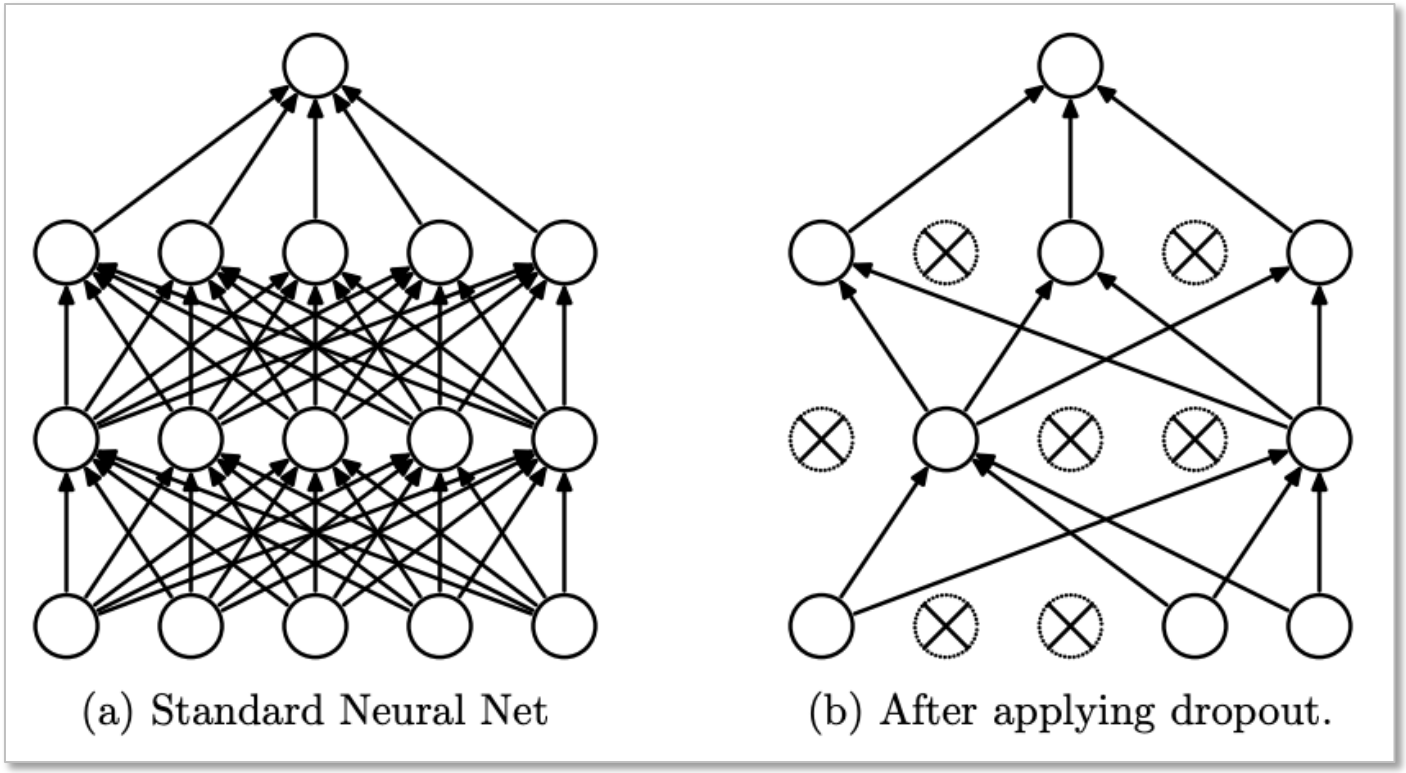
❖ 효과

- 앙상블(ensemble) 효과: 전체 데이터의 일부(mini-batch) 및 전체 특징의 일부에 대해 학습하고, 이를 무작위로 반복함으로써 앙상블 기법과 같이 모델을 일반화하는 효과를 가짐
- 동조화(co-adaptation) 방지: 학습이 진행되면서 모델에 속한 서로 다른 뉴런들이 강한 상관관계를 가지게 되어 불필요한 중복이 생기는 현상을 방지 → 강건(robust)한 모델을 구축할 수 있음

4. Dropout

개요

❖ 일반 인공신경망과 dropout이 적용된 신경망의 비교

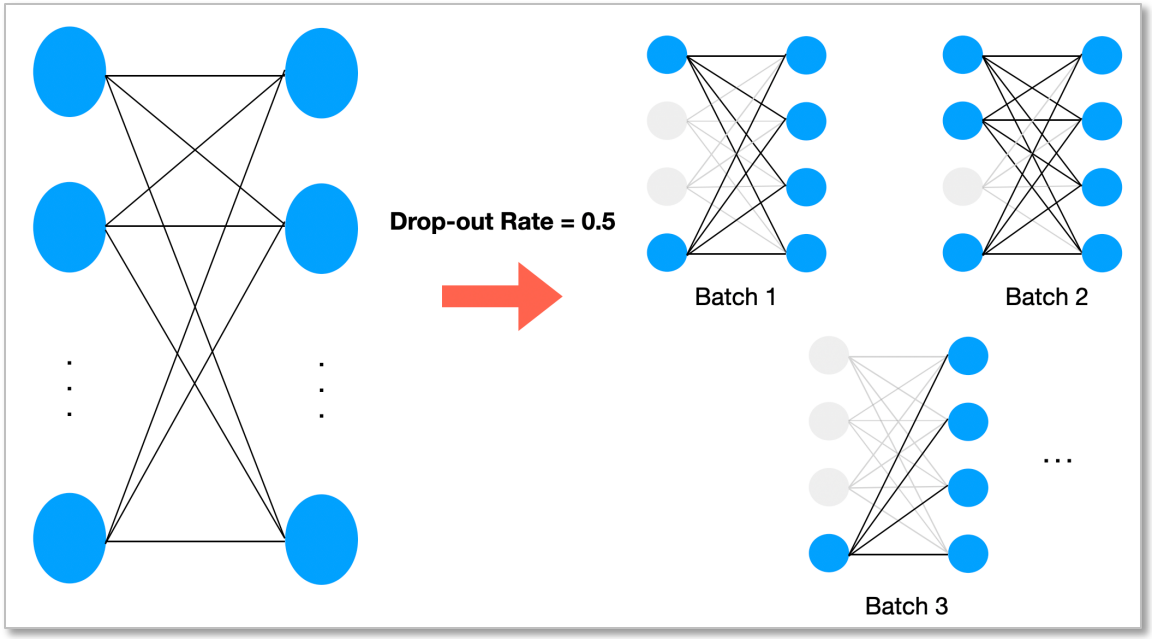


4. Dropout

구현 방식

❖ 모델 학습 과정

- 특정한 비율(dropout rate)에 따라 모델 학습 과정에서 각 mini-batch별로 활성화되는 뉴런을 다르게 적용하여 학습을 진행함
  - Dropout rate: 각 뉴런의 dropout 여부를 결정하는 확률. dropout rate=0.5라면 각 뉴런은 50% 확률로 활성화/비활성화됨

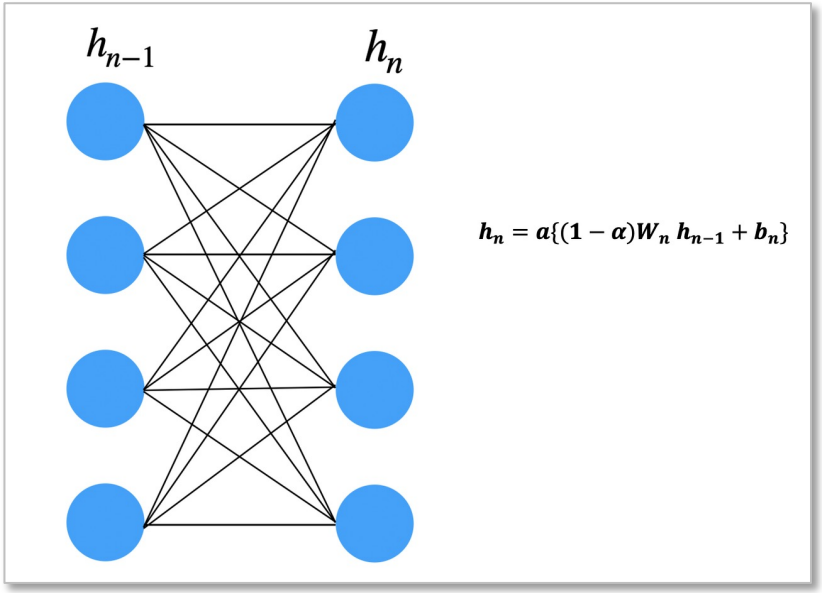


4. Dropout

구현 방식

❖ 모델 추론/검증 과정

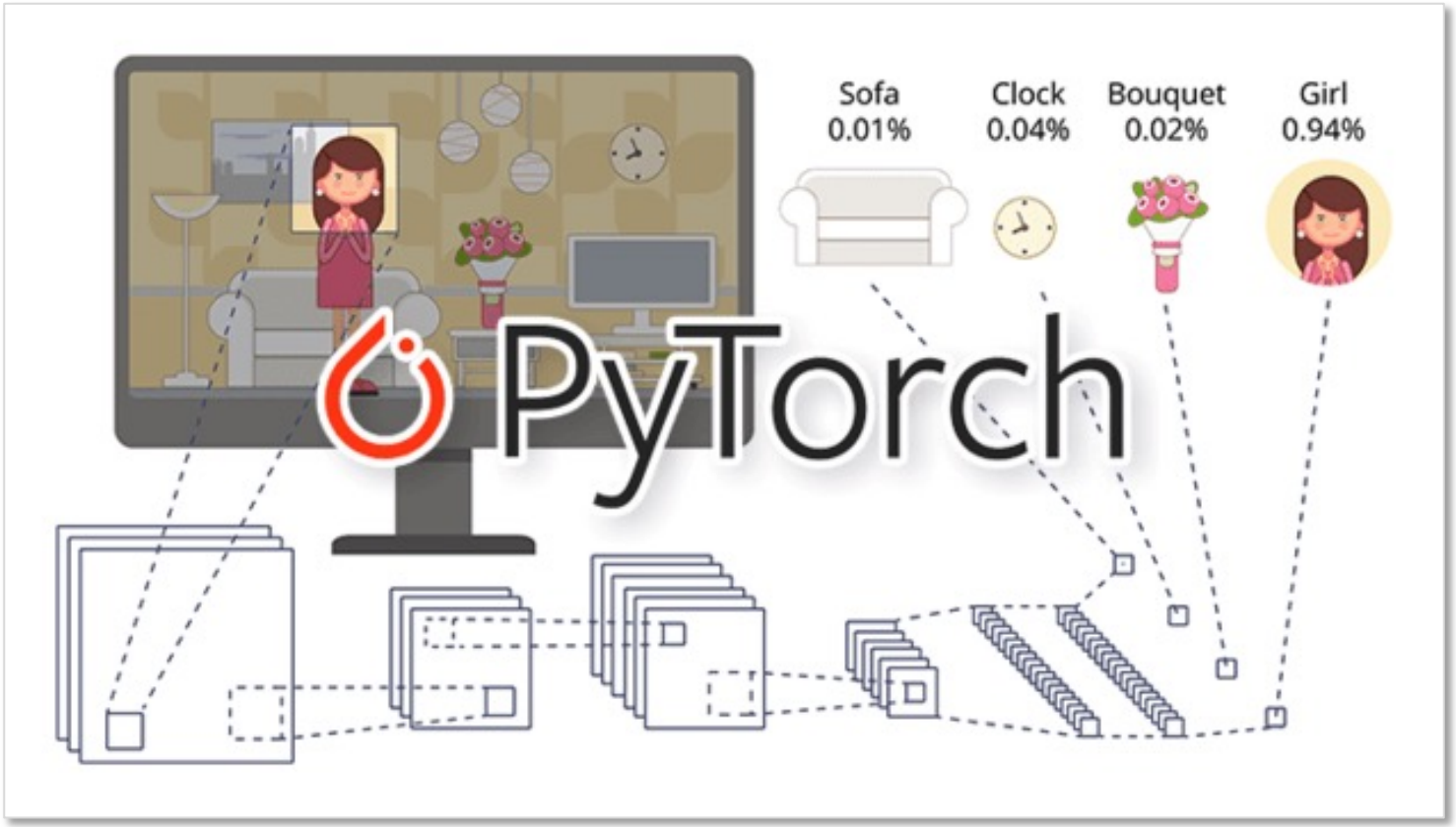
- 모델 학습이 완료된 후, 추론(inference) 또는 검증 시에는 모든 뉴런을 복원하여 사용
  - 모든 뉴런을 사용하게 되면, 학습 시 사용되던 뉴런의 수와 차이가 발생
    → 가중치의 크기를 조정(rescaling)하여 이를 보상
    - Dropout rate가 0.3인 경우, 매 학습 시 전체 가중치의 약 70%만 사용하여 값을 출력. 추론/검증 시 전체 가중치를 다 사용하게 되면 학습 때의 출력값보다 큰 값이 나올 수 있음
    - 전체 가중치에 (1-Dropout rate)을 곱하여 출력값을 조정



5. (실습) PyTorch

PyTorch

❖ 성능 개선 기법 적용(PyTorch)



Thank you

UNIST 융합경영대학원
이규민(Gyumin Lee)
glee.optimizt@gmail.com

ULSAN NATIONAL INSTITUTE OF
SCIENCE AND TECHNOLOGY