Chapter 9
Hyper-Parameter Tuning with Cross-Validation

YONSEI UNIVERSITY

CONTENTS

1. Motivation
2. Grid Search Cross-Validation
3. Randomized Search Cross-Validation
4. Scoring and Hyper-Parameter Tuning

01.

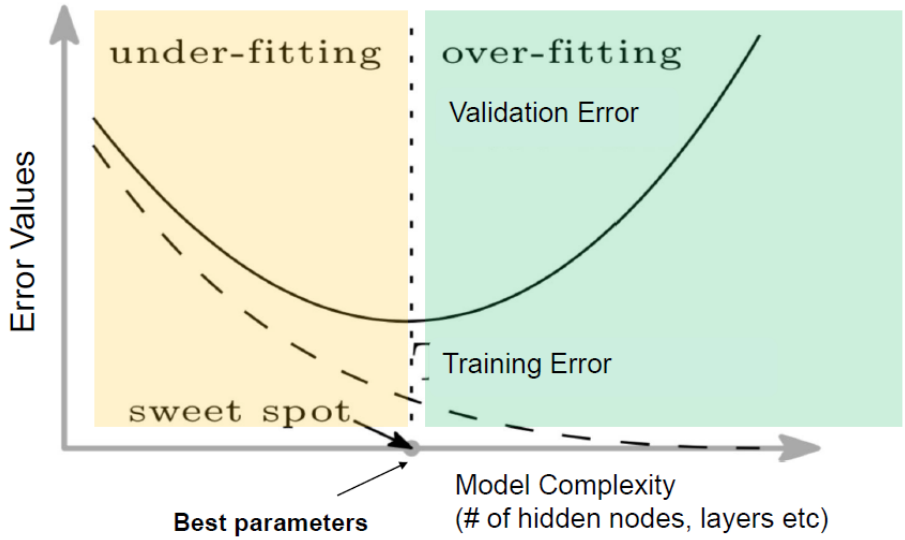
Motivation

1 Motivation

- **왜 하이퍼 파라미터 튜닝인가**
  - **계량경제 모델과 기계학습 모델의 목적과 철학**
    - MVUE
    - Minimize overall errors
  - Uncertainty
    - **Aleatoric Uncertainty**
      - 데이터 생성과정에서 발생하는 무작위성.
    - **Epistemic Uncertainty**
      - 정확한 모델의 파라미터를 모르기 때문에 발생하는 불확실성
    - **Out-of-Distribution Uncertainty**
      - 데이터가 원래 훈련 데이터 영역의 범위를 벗어난 경우

1 Motivation

- **하이퍼 파라미터 튜닝**
  - 적절히 이뤄지지 못할 경우 과적합 되기 쉬움
- **종류**
  - Grid Search Cross-Validation
  - Randomized Search Cross-Validation



02.

Grid Search Cross-Validation

2 Grid Search Cross-Validation

- **정의**
  - 완전 탐색을 통해 CV 성능 최대화 하이퍼 파라미터 조합 확인
- **특징**
  - 기저 데이터 구조를 잘 모를 경우 가장 먼저 취할 수 있는 방법
- **구현**
  - Scikit-learn의 GridSearchCV
    - 코드 9.1 퍼지 K-Fold 교차 검증을 사용한 그리드 탐색
      - 코드 7.3 관측값이 중첩될 때 교차-검증 클래스
      - 코드 9.2 보강된 PIPELINE 클래스

2 Grid Search Cross-Validation

- GridSearchCV
  - grid_scores_
    - Param_grid의 모든 조합에 대한 성능 결과
  - parameters
    - 사용된 파라미터
  - mean_validation_score
    - 교차 검증 결과의 평균값
  - cv_validation_scores
    - 모든 교차검증 결과
  - best_scores
    - 최고 점수
  - best_params
    - 최고 점수를 낸 파라미터
  - best_estimator_
    - 최고 점수를 낸 파라미터를 가진 모형

2 Grid Search Cross-Validation

9.1. 퍼지 K-kold 교차-검증을 사용한 그리드 탐색

```
def clfHyperFit(feat, lb1, t1, pipe_clf, param_grid, cv=3, bagging=[0,None,1.],
                n_jobs=-1, pctEmbargo=0, **fit_params):

    if set(lb1.values)=={0,1}:
        scoring='f1'#메타 레이블을 위한 f1(0이 대다수인 불균형 데이터)
    else:
        scoring='neg_log_loss' # (대칭인 경우)

    #1. 훈련 데이터에 대한 하이퍼라미터 튜닝
    inner_cv= PurgedKFold(n_splits=cv, t1=t1, pctEmbargo=pctEmbargo) # 퍼지
    gs = GridSearchCV(estimator=pipe_clf,param_grid=param_grid, scoring=scoring, cv=inner_cv, n_jobs=n_jobs, iid=False)
    gs = gs.fit(feat, lb1, **fit_params).best_estimator_

    #2. 데이터 전체에 대해 검증된 모델 적합화
    if bagging[1]>0:
        gs = BaggingClassifier(base_estimator=MyPipeline(gs.steps), n_estimators=int(bagging[0]),
                               max_samples=float(bagging[1]),max_features=float(bagging[2]), n_jobs=n_jobs)
        gs = gs.fit(feat, lb1, sample_weight=fit_params[gs.base_estimator.steps[-1][0]+'__sample_weight'])
        gs = Pipeline([('bag', gs)])
    return gs
```

2 Grid Search Cross-Validation

7.3. 관측값이 중첩될 때 교차-검증 클래스

```
from sklearn.model_selection._split import _BaseKFold
class PurgedKFold(_BaseKFold):
    '''
    K-폴드를 확장해 구간으로 확장된 레이블에 작동
    훈련은 테스트-레이블 구간과 중첩되는 관측값 퍼지
    테스트 집합은 런이라 가정'''
    '''
    n_split    : 분할 수
    t1         : Pandas Series
    pctEmbargo : 엠바고 비율(1%면 충분하나 데이터 수에 따라 변화)
    '''
    def __init__(self, n_splits=3, t1=None, pctEmbargo=0.):
        if not isinstance(t1, pd.Series):
            raise ValueError('must be pd series')
        super(PurgedKFold, self).__init__(n_splits, shuffle=False, random_state=None)
        self.t1=t1
        self.pctEmbargo=pctEmbargo
    def split(self, X, y=None, groups=None):
        if(X.index==self.t1.index).sum()!=len(self.t1):
            raise ValueError('X and thruDateValues must have the same index')
        indices = np.arange(X.shape[0])
        mbrg = int(X.shape[0]*self.pctEmbargo)
        test_starts=[(i[0], i[-1]+1) for i in np.array_split(np.arange(X.shape[0]), self.n_splits)]
        for i,j in test_starts:
            t0 = self.t1.index[i] #start of test set
            test_indices = indices[i:j]
            maxT1Idx = self.t1.index.searchsorted(self.t1[test_indices].max())
            train_indices = self.t1.index.searchsorted(self.t1[self.t1<=t0].index)
            train_indices = np.concatenate((train_indices, indices[maxT1Idx+mbrg:]))
            yield train_indices, test_indices
```

Grid Search Cross-Validation

9.2. 보강된 PIPELINE 클래스

```
class MyPipeline(Pipeline):
    def fit(self, X, y, sample_weight=None, **fit_params):
        if sample_weight is not None:
            fit_params[self.steps[-1][0]+'__sample_weight'] = sample_weight
        return super(MyPipeline, self).fit(X, y, **fit_params)
```


Thank you
YONSEI UNIVERSITY