deep learning optimization

cross validation & grid search & callbacks

https://youtu.be/h9ZeElQqwjY

HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

cross validation

❖ validation data을 사용하여 모델 성능 검증

- training 과정과 동일
- 목적

  새로운 데이터에 대한 성능 예측

  최적 모델 설계 (hyperparameter tuning 통해)

❖ Cross validation(교차 검증)

- 보통 training data set이 작은 경우 사용
- k-fold cross validation 주로 사용
  - 모든 data가 validation data로 한번씩 사용

    → 특정 dataset에 overfitting 방지
  - 모든 data가 training data로 한번씩 사용

    → 정확도 향상 & underfitting 방지
  - training, validation에 많은 시간 소요

test	training	training
training	test	training
training	training	test

각 경우의 정확도의 평균으로 최종 평가

k-fold cross validation

```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import KFold
model = KerasClassifier(build_fn = build_model, epochs=20, batch_size=20, verbose=2)
kfold =KFold(n_splits=10, shuffle=True, random_state=0)
results = cross_val_score(model,X_train,Y_train,cv=kfold)
```

몇 개의 세트로 나눌 것인지

모델 생성했던 함수명

각 dataset에 대해 epoch이 끝나면 정확도 산출

이 값들이 크게 차이나지 않는 경우 해당 모델을 선택

```
Epoch 20/20
6/6 - 0s - loss: 3.7167e-06 - acc: 1.0000
1/1 - 0s - loss: 1.0175 - acc: 0.9091
```

```
Epoch 20/20
6/6 - 0s - loss: 1.1785e-05 - acc: 1.0000
1/1 - 0s - loss: 1.3358 - acc: 0.8182
```

grid search

❖ **교차 검증을 통해 선택한 모델에 대해 최적 하이퍼 파라미터를 선택**

- 설정한 파라미터에 대해 모든 경우의 수로 학습하여 가장 좋은 하이퍼 파라미터를 선택하면 됨
- training 과정과 동일

```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
model = KerasClassifier(build_fn = build_model, epochs=20)#, batch_size=20, verbose=2)

epochs=[20,25] #20
#lr = [0.001, 0.01, 0.1, 0.2, 0.3]
#batch_size=[5,10] #5
steps_per_epoch = [10,20] # 10

param_grid = dict(epochs = epochs, steps_per_epoch=steps_per_epoch)
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=5)
grid_result = grid.fit(X_train, Y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

구성해놓은 모델

검색하고자 하는 하이퍼 파라미터를 설정

학습과정과 동일 (model.fit 대신 grid.fit)

```
Best: 0.830040 using {'epochs': 20, 'steps_per_epoch': 10}
0.830040 (0.075916) with: {'epochs': 20, 'steps_per_epoch': 10}
nan (nan) with: {'epochs': 20, 'steps_per_epoch': 20}
0.830040 (0.052332) with: {'epochs': 25, 'steps_per_epoch': 10}
0.812648 (0.050732) with: {'epochs': 25, 'steps_per_epoch': 20}
```

call back

- **callback**

  deep learning에서 epoch에 따른 overfitting, underfitting을 방지하기 위해 epoch을 많이 돌린 후 특정 지점에서 중지하는 방법

- **Earlystopping 클래스 사용**

  구성 요소 : performance measure(모니터링할 성능) & trigger(학습을 멈출 기준)

```
early_stop = EarlyStopping(monitor='val_loss', patience=5, verbose=1)

early_stop = EarlyStopping(monitor='val_loss', mode='min', baseline=0.4)
```

- **patience** : 성능이 증가하지 않는 epoch을 몇 번 허용할 것인지
- **baseline** : 특정 값에 도달하였을 때 중지
- **min / max** : 모니터링하고 있는 값이 최소가 될 때 / 최대가 될 때 중지

```
model.fit(X_train, y_train, epochs=100, batch_size = 30,  verbose=1 , callbacks=[early_stop])
```

callbacks에 early_stop 객체를 넣어주어 적용

Q & A