기계학습 활용 (12주차)

2019. 11. 29.

Prof. Seung Ho Lee

KOREATECH
한국기술교육대학교

강의 주제 : 딥러닝 활용(과적합 피하기)

- 이론
  - 과적합 이해
  - K겹 교차 검증

- 실습
  - 과적합 이해 / K겹 교차 검증 (초음파로 광물 예측하기)

지난 강의 요약

- 지난 강의의 딥러닝 모델 평가
  - 학습용 데이터와 테스트용 데이터가 구분되지 않았음

  - 예 : 폐암 환자 생존여부 예측 문제
    - ✓ 딥러닝 모델 학습에 사용한 기존 환자 데이터 중 일부를 랜덤 추출
    - ✓ 랜덤 추출 데이터를 새 환자인 것으로 가정하고 평가(테스트) 수행

출처 : 모두의 딥러닝(조태호)

오늘 강의 요약

- 목표
  - 앞에 언급한 방법으로 딥러닝 모델 평가 시 신뢰성이 떨어지는 문제점(과적합) 이해
  - 내가 보유한 전체 데이터 셋을 학습용과 테스트용으로 구분하는 개념 이해
  - 테스트용 데이터를 최대한 확보하기 위한 방법(K겹 교차 검증) 이해

출처 : 모두의 딥러닝(조태호)

딥러닝 문제 정의

- 초음파 광물 예측
  - 1988년 존스 홉킨스 대학의 세즈노프스키 교수는 2년 전 힌튼 교수가 발표한 역전파 알고리즘에 관심을 가짐
  - 그는 은닉층과 역전파 알고리즘이 얼마나 효과가 있는지 직접 실험해보기 위해 광석과 일반 돌을 놓고 음파 탐지기를 쏜 후 결과를 데이터로 정리하였음
  - 오차 역전파 알고리즘을 사용한 신경망이 광석과 돌을 구분하는데 얼마나 효과적인지 확인해보자

출처 : 모두의 딥러닝(조태호)

딥러닝 문제 정의

- 데이터 셋 확인

Connectionist Bench (Sonar, Mines vs. Rocks) Data Set

Download: Data Folder, Data Set Description

Abstract: The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock.

Data Set Characteristics:	Multivariate	**Number of Instances:**	208	**Area:**	Physical
Attribute Characteristics:	Real	**Number of Attributes:**	60	**Date Donated**	N/A
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	159117

Data Set Information:

The file "sonar.mines" contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions. The file "sonar.rocks" contains 97 patterns obtained from rocks under similar conditions. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency. The data set contains signals obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock.

Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occur later in time, since these frequencies are transmitted later during the chirp.

The label associated with each record contains the letter "R" if the object is a rock and "M" if it is a mine (metal cylinder). The numbers in the labels are in increasing order of aspect angle, but they do not encode the angle directly.

출처 :
http://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks)

딥러닝 문제 정의

- 데이터 셋 확인
  - 속성 개수 : 60개 (0~59열)
  - 클래스 : 일반 돌R 또는 광석M (60번째 열)
  - 총 샘플의 개수 : 208개

	0	1	2	3	...	59	60
0	0.02	0.0371	0.0428	0.0207	...	0.0032	R
1	0.0453	0.0523	0.0843	0.0689	...	0.0044	R
2	0.0262	0.0582	0.1099	0.1083	...	0.0078	R
3	0.01	0.0171	0.0623	0.0205	...	0.0117	R
4	0.0762	0.0666	0.0481	0.0394	...	0.0094	R

출처 : 모두의 딥러닝(조태호)

딥러닝 문제 정의

- 실험결과
  - 모델 정확도 : 99.52%

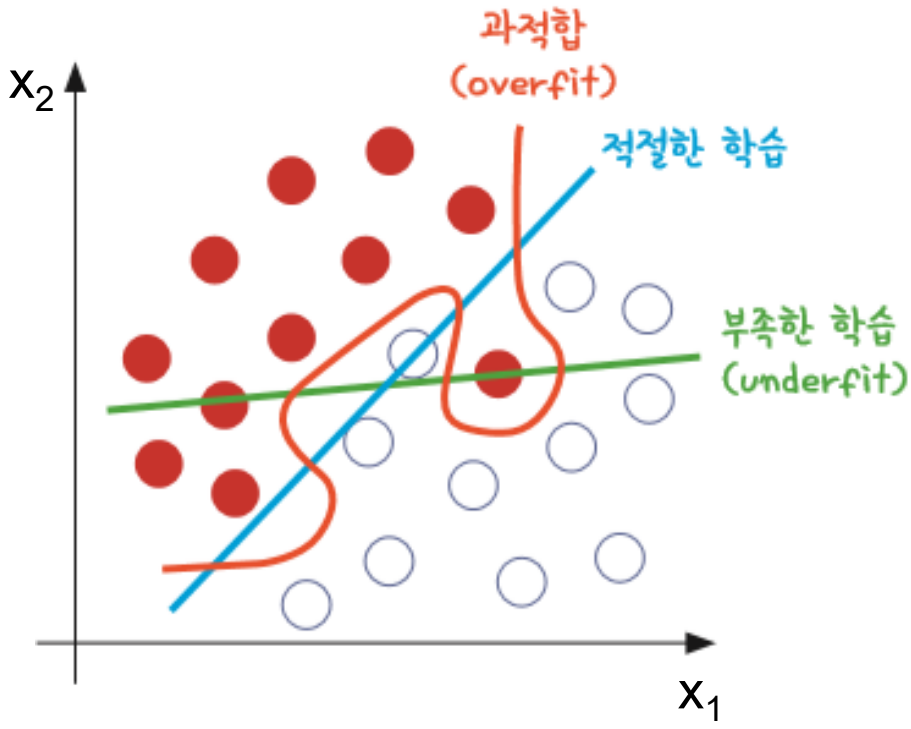
```
Epoch 1/200
208/208 [==============================] - 0s - loss: 0.2532 - acc: 0.4567
Epoch 2/200
208/208 [==============================] - 0s - loss: 0.2457 - acc: 0.5385
Epoch 3/200
208/208 [==============================] - 0s - loss: 0.2426 - acc: 0.5385

(중략)

Epoch 198/200
208/208 [==============================] - 0s - loss: 0.0057 - acc: 0.9952
Epoch 199/200
208/208 [==============================] - 0s - loss: 0.0057 - acc: 0.9952
Epoch 200/200
208/208 [==============================] - 0s - loss: 0.0057 - acc: 0.9952
 32/208 [===>..........................] - ETA: 0s
Accuracy: 0.9952
```


출처 : 모두의 딥러닝(조태호)

과적합

- **과적합**Overfitting
  - **모델이 학습 데이터 셋에 적용하면 높은 정확도를 보이지만 새로운 데이터에 적용하면 정확도가 떨어지는 현상**

- 그래프에서 빨간색을 보면 주어진 샘플에 정확히 맞게끔 선이 그어져 있음
- 하지만 이 선은 너무 이 경우에만 최적화되어 있음
- 다시 말해서, 완전히 새로운 데이터에 적용하면 이 선을 통해 정확히 두 그룹을 나누지 못하게 된다는 의미



출처 : 모두의 딥러닝(조태호)

과적합

- 과적합이 발생하는 경우
  - 신경망 층 수가 너무 많거나 복잡한 경우
    - ✓ 특히 딥러닝은 학습 단계에서 은닉층, 출력층의 노드 들에 매우 많은 변수들이 투입됨
  - 테스트 셋이 학습용 셋과 중복되는 경우
    - ✓ 예 : 딥러닝 모델 학습에 사용한 데이터를 그대로 모델 평가에 사용한 경우
    - ✓ 중복인 경우와 중복이 안 되는 경우의 정확도 차이를 알아볼 것

출처 : 모두의 딥러닝(조태호)

학습 셋과 테스트 셋

- 과적합을 방지하기 위한 방법
  - 학습을 하기 위한 데이터 셋과 테스트 할 데이터 셋을 완전히 구분한 다음 학습과 동시에 평가를 병행하는 것이 한 방법

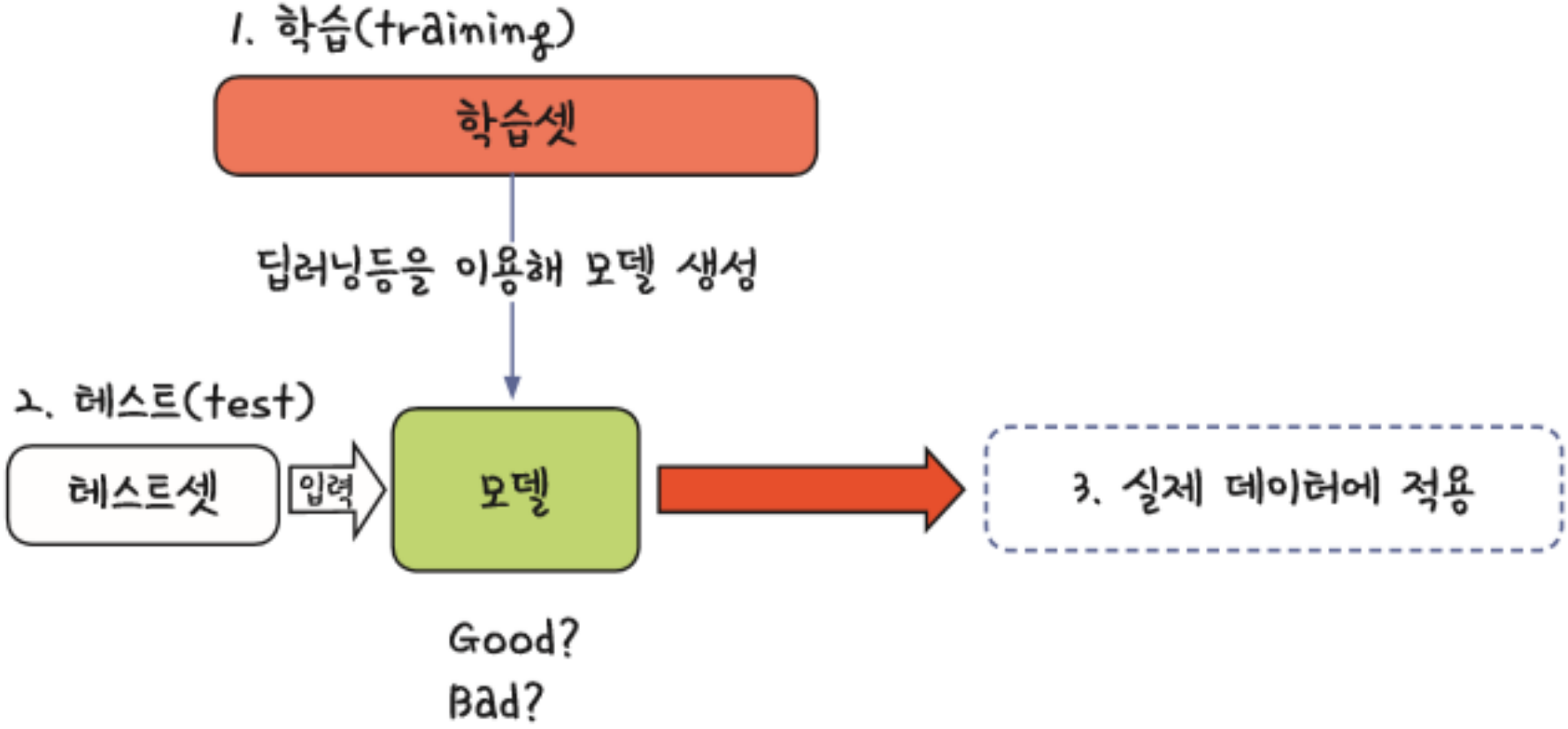
  - 예 : 데이터 셋이 총 100개의 샘플로 이루어진 경우 다음과 같이 두 개의 분리된 셋으로 나눌 수 있음

70개 샘플은 학습셋으로	30개 샘플은 테스트셋으로

출처 : 모두의 딥러닝(조태호)

학습 셋과 테스트 셋

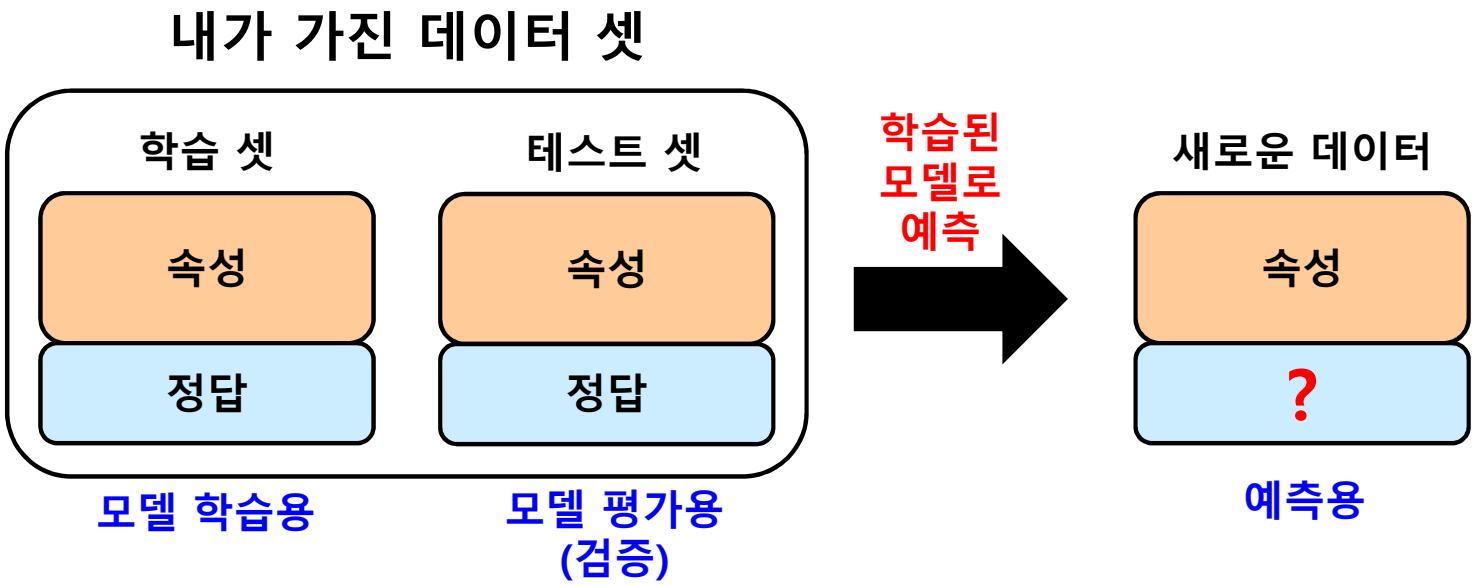
- 모델 평가 및 예측 순서



출처 : 모두의 딥러닝(조태호)

학습 셋과 테스트 셋

- 전체 데이터 셋의 구성



출처 : 모두의 딥러닝(조태호)

학습 셋과 테스트 셋

- 과적합의 발생
  - 학습 셋만 가지고 평가할 때 은닉층을 추가하거나 에포크 값을 높여 반복횟수를 늘리면 정확도가 계속 올라갈 수 있음
  - 하지만 학습 셋에 대해서만 평가한 정확도가 테스트 셋에서도 그대로 유지되기는 어려움

출처 : 모두의 딥러닝(조태호)

학습 셋과 테스트 셋

- 과적합의 발생
  - 세즈노프스키 교수의 실험 데이터 일부
    - ✓ 학습 셋의 경우 은닉층 수가 많을수록 예측률 증가
    - ✓ 테스트 셋은 은닉층 수가 12개일 때 90.4%로 최고를 기록하고 24개일 때에는 다시 89.2%로 떨어짐

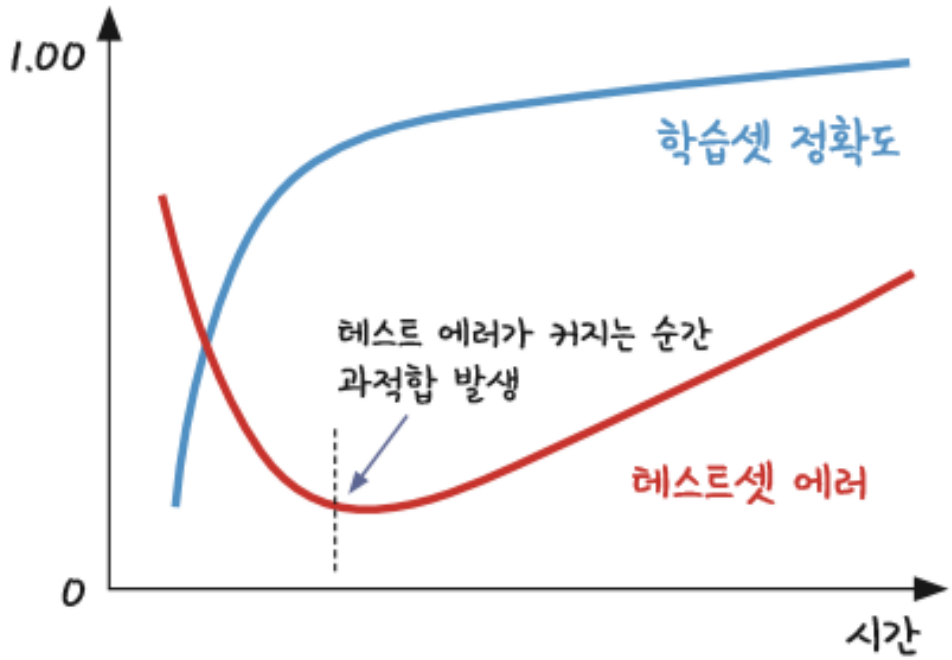
➔ 즉, 신경망이 복잡해질수록 학습 셋에 대한 성능은 높아지지만 테스트 셋에 대한 성능은 오히려 떨어지는 경우가 존재

은닉층 수의 변화	학습셋의 예측률	테스트셋의 예측률
0	79.3	73.1
2	96.2	85.7
3	98.1	87.6
6	99.4	89.3
12	99.8	90.4
24	100	89.2

출처 : 모두의 딥러닝(조태호)

학습 셋과 테스트 셋

- **과적합의 발생**
  - 학습을 반복적으로 수행할 때 테스트 셋에 대한 정확도가 더 이상 높아지지 않는 지점에서 학습을 멈춰야 함
    - ✓ 이 지점에서의 학습 정도가 가장 적합한 것으로 볼 수 있음



출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : 기본 소스코드

- 기본 소스코드 (학습 셋으로 모델 평가)

```
# 라이브러리 불러오기
from keras.models import Sequential
from keras.layers.core import Dense
from sklearn.preprocessing import LabelEncoder

import pandas as pd
import numpy
import tensorflow as tf

# seed 값 설정
seed = 0
numpy.random.seed(seed)
tf.set_random_seed(seed)

df = pd.read_csv('./dataset/sonar.csv', header=None)
```


출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : 기본 소스코드

```
# 데이터 입력
dataset = df.values
X = dataset[:,0:60]
Y_obj = dataset[:,60]

# 클래스 문자열(R,M)→숫자(0,1) 변환
e = LabelEncoder()
e.fit(Y_obj)
Y = e.transform(Y_obj)

# 신경망 모델 설정
model = Sequential()
model.add(Dense(24, input_dim=60, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```


출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : 기본 소스코드

```
# 모델 컴파일
model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=['accuracy'])

# 모델 실행
model.fit(X, Y, epochs=130, batch_size=5)

# 결과 출력
print("\n Accuracy: %.4f" % (model.evaluate(X, Y)[1]))
```

- 실행 결과
  - ✓ 98.56% : 예측 정확도
  - ✓ 학습 셋을 모델 평가에 사용한 결과

출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : 학습 셋 테스트 셋 구분

- 주어진 데이터를 학습 셋과 테스트 셋으로 나누어 평가하는 예제
- 불러온 X, Y 데이터를 정해진 비율(%)만큼 구분하여 한 그룹은 학습에 사용하고 다른 한 그룹은 테스트에 사용
  - ✓ Sklearn 라이브러리의 train_test_split() 함수 사용

```
from sklearn.model_selection import train_test_split

# 학습셋과 테스트셋의 구분(테스트 셋 30%, 학습 셋 70%)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,
random_state=seed)
```


출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : 학습 셋 테스트 셋 구분

- 모델 학습에서는 학습 셋 사용
  - ✓ X_train : 학습 셋의 속성
  - ✓ Y_train : 학습 셋의 클래스

- 모델 평가에서는 테스트 셋 사용
  - ✓ X_test : 테스트 셋의 속성
  - ✓ Y_test : 테스트 셋의 클래스

```
# 모델 학습 : 학습 셋 사용
model.fit(X_train, Y_train, epochs=130, batch_size=5)

# 모델 평가 : 테스트 셋 사용
print("\n Test Accuracy: %.4f" % (model.evaluate(X_test, Y_test)[1]))
```

초음파 광물 예측 : 학습 셋 테스트 셋 구분

- 학습 셋 테스트 셋 구분 적용 소스코드

```
# 라이브러리 불러오기
from keras.models import Sequential
from keras.layers.core import Dense
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

import pandas as pd
import numpy
import tensorflow as tf

# seed 값 설정
seed = 0
numpy.random.seed(seed)
tf.set_random_seed(seed)
```

코드 추가

출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : 학습 셋 테스트 셋 구분

```
# 데이터 입력
df = pd.read_csv('./dataset/sonar.csv', header=None)

dataset = df.values
X = dataset[:,0:60]
Y_obj = dataset[:,60]

# 클래스 문자열(R,M)→숫자(0,1) 변환
e = LabelEncoder()
e.fit(Y_obj)
Y = e.transform(Y_obj)

# 학습셋과 테스트셋의 구분
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=seed)

# 신경망 모델 설정
model = Sequential()
model.add(Dense(24, input_dim=60, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

코드 추가

출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : 학습 셋 테스트 셋 구분

```
# 모델 컴파일
model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=['accuracy'])

# 모델 실행                                    X        Y
model.fit(X_train, Y_train, epochs=130, batch_size=5)

# 결과 출력                                                X        Y
print("\n Accuracy: %.4f" % (model.evaluate(X_test, Y_test)[1]))
```

코드 수정

코드 수정

- 실행 결과
  - ✓ 80.95% : 예측 정확도
  - ✓ 학습 셋으로 실험했을 때의 98.56%보다 확연히 낮은 정확도를 보임

출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : 모델 저장과 재사용

- 학습 후에 테스트 결과가 만족스러울 때 이를 모델로 저장하여 새로운 데이터에 사용할 수 있음

```
# 모델 컴파일
model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=['accuracy'])

# 모델 실행
model.fit(X_train, Y_train, epochs=130, batch_size=5)

from keras.models import load_model  # 모델 불러오기 라이브러리 추가
model.save('my_model.h5')   # 모델을 컴퓨터에 저장

del model       # 테스트를 위해 메모리 내의 모델을 삭제
model = load_model('my_model.h5') # 모델을 새로 불러옴

# 결과 출력
print("\n Accuracy: %.4f" % (model.evaluate(X_test, Y_test)[1]))
```

<학습 셋
테스트 셋 구분>
소스코드에 추가

출처 : 모두의 딥러닝(조태호)

테스트 셋 부족 문제

- 테스트 셋 부족 문제
  - 내가 가진 데이터 셋의 약 70%를 학습 셋으로 사용하여 테스트 셋은 겨우 30%에 그침
  - 이 정도의 테스트 셋으로는 딥러닝 모델이 실제로 얼마나 잘 동작하는 지 파악하기가 어려움

출처 : 모두의 딥러닝(조태호)

K겹 교차 검증

- **K겹 교차 검증**k-fold cross validation
  - 학습 셋/테스트 셋 구성을 여러 가지로 두고 하나씩 성능을 평가하여 종합(=평균)하는 방법
  - 데이터 셋의 100%를 테스트 셋으로 활용할 수 있음

5겹 교차 검증의 예시

출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : K겹 교차 검증

- K겹 교차 검증 소스코드
  - ✓ 〈학습 셋 테스트 셋 구분〉 소스코드를 불러와서 아래와 같이 수정해보자

빨간색 박스 부분만 수정

```
from keras.models import Sequential
from keras.layers.core import Dense
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold # 데이터를 k겹으로 구성할 수 있도록 해주는 라이브러리

import numpy
import pandas as pd
import tensorflow as tf

# seed 값 설정
seed = 0
numpy.random.seed(seed)
tf.set_random_seed(seed)

df = pd.read_csv('./dataset/sonar.csv', header=None)
```


출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : K겹 교차 검증

빨간색 박스
부분만 수정

```
dataset = df.values
X = dataset[:,0:60]
Y_obj = dataset[:,60]

e = LabelEncoder()
e.fit(Y_obj)
Y = e.transform(Y_obj)

# 10개의 파일로 쪼갬
K = 10
kFold = StratifiedKFold(n_splits=K, shuffle=True, random_state=seed) # 데이터 k겹 구성

# 빈 accuracy 배열 생성(10개의 정확도를 담을 배열)
accuracy = []
```


출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : K겹 교차 검증

```
# 모델의 설정, 컴파일, 실행
for train_index, test_index in kFold.split(X, Y):
    model = Sequential()
    model.add(Dense(24, input_dim=60, activation='relu'))
    model.add(Dense(10, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='mean_squared_error',
                  optimizer='adam',
                  metrics=['accuracy'])
    model.fit(X[train_index], Y[train_index], epochs=130, batch_size=5)
    k_accuracy = "%.4f" % (model.evaluate(X[test_index], Y[test_index])[1]) # 테스트 1번 정확도
    accuracy.append(k_accuracy) # k_accuracy를 순차적으로 accuracy 배열에 추가적으로 담기

# 결과 출력
print("\n %.f fold accuracy:" % K, accuracy) # 10번의 테스트 결과를 출력
```

모델을 설정하고 실행하는 부분을 반복문 (for 문)으로 묶어 k-fold(k=10)만큼 반복

출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : K겹 교차 검증

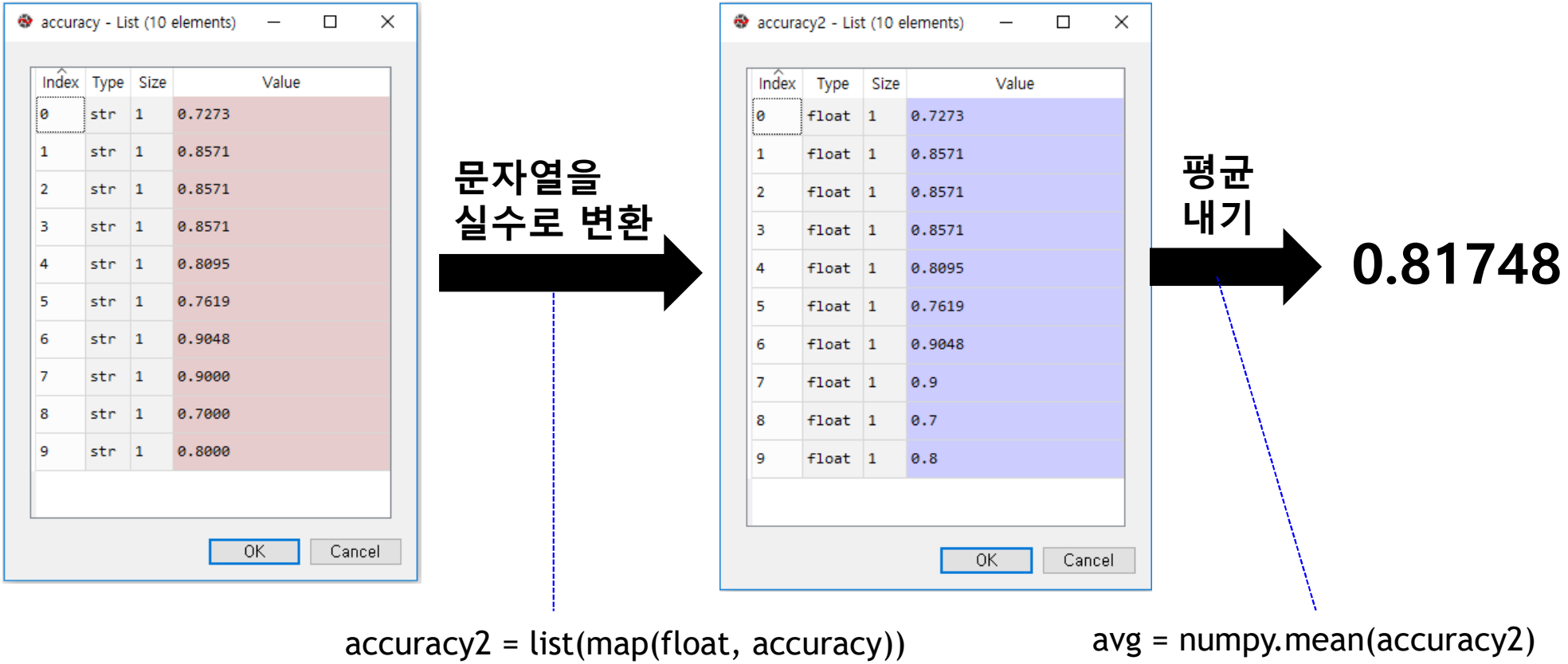
- 실행 결과
  - ✓ 10번의 테스트 결과가 한꺼번에 출력되는 것 확인

```
(중략)

Epoch 128/130
188/188 [==============================] - 0s 416us/step - loss: 0.0107 - acc: 1.0000
Epoch 129/130
188/188 [==============================] - 0s 415us/step - loss: 0.0115 - acc: 1.0000
Epoch 130/130
188/188 [==============================] - 0s 499us/step - loss: 0.0124 - acc: 0.9947
20/20 [==============================] - 0s 15ms/step

 10 fold accuracy: ['0.7273', '0.8571', '0.8571', '0.8571', '0.8095', '0.7619', '0.9048', '0.9000',
'0.7000', '0.8000']
```


출처 : 모두의 딥러닝(조태호)

초음파 광물 예측 : K겹 교차 검증

- 실행 결과
  - ✓ 10번의 테스트 결과를 평균 내보기



accuracy2 = list(map(float, accuracy))

avg = numpy.mean(accuracy2)

출처 : 모두의 딥러닝(조태호)