

< 도시소리 분류하기 >

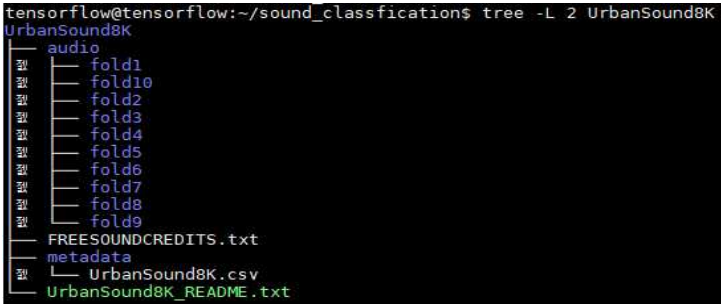
1. 실습내용 요약

- 딥러닝을 활용하여 도시의 소리를 구분(classification)하고자 함.
- 참고 : <https://medium.com/@mikesmales/sound-classification-using-deep-learning-8bc2aa1990b7>
<http://keunwoochoi.blogspot.com/2016/01/blog-post.html>
- full 코드 : <https://github.com/mikesmales/Udacity-ML-Capstone>

2. 실습자료 준비

- urban sound 다운로드(<https://urbansounddataset.weebly.com/urbansound8k.html>)
- jupyter notebook

3. 실습시작

구 분	내 용						
① 다운로드 받은 sound 파일 압축풀기	<p>\$ tar -zxvf UrbanSound8K.tar.gz → z(tar.gz 파일을) x(extract 압축해제하라) v(verbosely 과정을 다 보이게) f(file의 이름은 다음과 같음) (참고 : https://brownbears.tistory.com/161)</p>  <p>→</p> <p>* 총 10개 class에서 8732개의 urban sound .wav file(<=4s)이 존재 * 에어컨, 자동차 경적, 어린이들 노는 소리, 개 짖는 소리, 드릴소리 등</p>						
② data 분석 및 시각화 (오디오의 속성 확인) (★용어정리!!★)	<p>\$ pip3 install librosa → python에서 많이 쓰이는 음성파일 분석 모듈 https://anaconda.org/conda-forge/librosa</p> <p>\$ pip3 install ffmpeg → librosa 사용을 위해 반드시 필요</p> <p>사람은 파형이 비슷한 소리도 구분 할 수 있지만 과연 딥러닝을 통해 기계를 학습시켰을 때 비슷하게 구분할 수 있을지가 관건!</p> <table border="1"> <tr> <td>Audio channel</td><td>Mono(모노)와 Stereo(스테레오)로 구분 참고: https://anaconda.org/conda-forge/librosa Mono = 1개 채널, AM / Stereo = 2개 채널, FM</td></tr> <tr> <td>Sample rate (자름)</td><td>초당 sampling 횟수로 단위는 Hz(주파수 정보) 참고: http://lgaeon.com/Awrm1tgh-Hjncd8gN-20701674</td></tr> <tr> <td>Bit depth (— 자름)</td><td>오디오의 음량 총량을 담는 컨테이너 참고: http://lgaeon.com/Awrm1tgh-Hjncd8gN-20701674</td></tr> </table>	Audio channel	Mono(모노)와 Stereo(스테레오)로 구분 참고: https://anaconda.org/conda-forge/librosa Mono = 1개 채널, AM / Stereo = 2개 채널, FM	Sample rate (자름)	초당 sampling 횟수로 단위는 Hz(주파수 정보) 참고: http://lgaeon.com/Awrm1tgh-Hjncd8gN-20701674	Bit depth (— 자름)	오디오의 음량 총량을 담는 컨테이너 참고: http://lgaeon.com/Awrm1tgh-Hjncd8gN-20701674
Audio channel	Mono(모노)와 Stereo(스테레오)로 구분 참고: https://anaconda.org/conda-forge/librosa Mono = 1개 채널, AM / Stereo = 2개 채널, FM						
Sample rate (자름)	초당 sampling 횟수로 단위는 Hz(주파수 정보) 참고: http://lgaeon.com/Awrm1tgh-Hjncd8gN-20701674						
Bit depth (— 자름)	오디오의 음량 총량을 담는 컨테이너 참고: http://lgaeon.com/Awrm1tgh-Hjncd8gN-20701674						

3. 실습시작(이어서)

구 분	내 용
③ data 전처리 및 구분	<p>pandas.Series.value_counts() 함수에서 normalize 인자의 기본값은 FALSE(개수)인데, TRUE로 주게 되면 상대적 비율 구함.</p> <p>normalization = 정규화(많은 양의 데이터를 처리할 때 데이터의 범위를 일치시키거나 분포를 유사하게 만들어주는 작업) (참고 : https://brunch.co.kr/@andrewhwan/16)</p> <p>많은 전처리 과정에서 Librosa의 load() 함수를 사용 기본적으로 librosa는 샘플링 속도를 22.05KHz로 변환해 비교 librosa의 출력을 scipy의 wavfile 라이브러리의 출력과 비교 (참고 : https://librosa.github.io/librosa/generated/librosa.core.load.html https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.io.wavfile.read.html)</p> <p>MFCC(Mel-Frequency Cepstral Coefficients) 음성인식의 Feature로 많이 사용되는 MFCC는 입력된 소리 전체를 대상으로 하는 것이 아니라, 일정 시간(구간)으로 나누어 이 시간에 대한 스펙트럼을 분석해 특징을 추출하는 기술. 소리의 고유한 특징을 나타내는 수치. (참고 : https://brightwon.tistory.com/11, http://blog.aladin.co.kr/2085738pyRe/1527444000, http://wikiscience.tistory.com/entry/%EC%9D%8C%EC%84%B1%EC%9D%B8%EC%8B%8D)</p> <p>librosa의 mfcc() 함수 활용 (https://librosa.github.io/librosa/generated/librosa.feature.mfcc.html)</p> <p>.csv 파일의 class_name(→class_label)이 문제의 답(label)</p> <p>sklearn(사이킷런) 모듈을 통해 전처리 카테고리 text 데이터를 기계가 알아들을 수 있게 숫자로 변환 label 16개를 one-hot-encoding 배열로 만들어 정답인 칸만 1이 되도록 변환(아래 그림과 같은 결과 도출)</p> <pre> features class_label ['dog_bark' 'children_playing' 'children_playing' ... 'car_horn' 'car_horn' 'car_horn'] Encoding features class_label [[0. 0. 0. ... 0. 0. 0.] [0. 0. 1. ... 0. 0. 0.] [0. 0. 1. ... 0. 0. 0.] ... [0. 1. 0. ... 0. 0. 0.] [0. 1. 0. ... 0. 0. 0.] [0. 1. 0. ... 0. 0. 0.]] </pre> <p>jupyter notebook의 자주 사용하는 Magic Command를 통해 이번 단계에서 전처리한 데이터를 모두 저장(%store ~~) (참고 : https://sosomemo.tistory.com/60)</p>

3. 실습시작(이어서)

앞서 ③에서 저장한 데이터를 (%store -r) 불러와서 실습시작
keras를 통해 Multilayer Perceptron(MLP) 신경망 구축 시작
3개의 레이어(입력, 숨겨진, 출력)

입력층	40개의 MFCC(columns)를 포함하는 1x40 배열
숨겨진 층	256개의 노드, 활성화함수는 relu 사용, 50% 드롭아웃 (참고 : https://pythonkim.tistory.com/40)
출력층	드롭아웃을 통해 노드를 랜덤으로 배제하여 더 나은 일반화를 하고(과적합률 ↓) label의 수와 일치하는 10개의 노드로 출력(활성화 함수는 softmax 사용)

④ 모델 학습(훈련) 및 평가(classification)

```
num_labels = yy.shape[1] # 10
filter_size = 2

# Construct model
model = keras.Sequential([
    keras.layers.Dense(256, activation='relu', input_shape=(40,)),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(num_labels, activation='softmax')
])

In [10]: # 모델 컴파일 하기
# 손실함수는 classification에 가장 많이 쓰이는 categorical_crossentropy 사용(점수 낮으
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

In [11]: model.summary() # 모델의 architecture summary

# pre-training 정확도 계산하기
score = model.evaluate(x_test, y_test, verbose=0)
accuracy = 100*score[1]

print("Pre-training accuracy: %.4f%%" % accuracy)

Model: "sequential_1"
-----
Layer (type)                 Output Shape              Param #
-----
dense_3 (Dense)              (None, 256)               10496
dropout_2 (Dropout)          (None, 256)               0
dense_4 (Dense)              (None, 256)               65792
dropout_3 (Dropout)          (None, 256)               0
dense_5 (Dense)              (None, 10)                2570
-----
Total params: 78,858
Trainable params: 78,858
Non-trainable params: 0
Pre-training accuracy: 10.7613%
```

Training은 epoch(반복횟수) 100회, batch size는 작게 설정
학습에 걸리는 시간이 많이 소요되므로 향후 재사용을 위해
callback 함수로 tf.keras.callbacks.ModelCheckpoint()를 통해
체크포인트 저장하기!

```
checkpoint_path = './saved_models/weights.best.basic_mlp.hdf5'

# 학습(Training)하는 동안 체크포인트 저장하기
# 훈련 중반과 마지막에 자동으로 저장하도록 옵션 설정(모델 재사용성 up!)
checkpointer = tf.keras.callbacks.ModelCheckpoint(checkpoint_path,
                                                  verbose=1, save_best_only=True)
```

model.predict() 함수를 통해 예측해보기 => 거의 일치!!

3. 실습시작(이어서)

<p>⑤ 모델 강화학습 (Refinement)</p>	<p>추출한 특징 MFCC 벡터의 크기는 오디오 파일마다 다르기 때문에 동일한 크기로 만들어줘야 함.</p> <p>모델을 CNN(Convolutional Neural Network)으로 수정.</p> <p>Convolutional(나선형의) layer는 특징 검출을 위해 설계.</p> <p>첫 번째 계층은 입력모양이 (40, 174, 1)인데 40은 MFCC의 숫자, 174는 패딩을 고려한 프레임 수, 1은 오디오가 Mono임을 의미!</p> <p>활성화함수는 이전 모델과 동일하게 ReLU 모델 사용.</p> <p>출력계층은 labels과 동일하게 10개의 노드로 설정.</p> <p>※ batch_size는 학습할 때 사용되는 데이터 수를 의미</p>
<p>⑥ 오디오 샘플링하기 (.wav 파일에서 음파를 1차원 벡터로 변환)</p>	<p>아래 영상 참고(샘플링에 대한 개념 설명)</p> <p>https://www.youtube.com/watch?time_continue=3&v=yWqpx08UeUs&feature=emb_logo</p> <p>* Aliasing = 디지털 신호 처리 과정에서 발생하는 노이즈(소리가 우둘투둘한 것)</p>
<p>참 고</p>	<p>현업에서 많이 사용하는 python 모듈 1위? Pandas / 2위? sklearn 머신러닝 라이브러리인 sklearn(사이킷런)은 지도·비지도학습 모듈, 모델 선택 및 평가모듈, 데이터 변환 및 불러오기 위한 모듈, 계산 성능 향상을 위한 모듈을 기본으로 하고 있음.</p> <p>대부분 통일된 인터페이스를 가지고 있어 여러 기법 적용에 용이 [그림 참고 : https://scikit-learn.org/stable/ 사이킷런 알고리즘]</p> <div data-bbox="507 1375 1452 1957"> <p>The flowchart 'scikit-learn algorithm cheat-sheet' provides a systematic approach to choosing an algorithm. It begins at a 'START' node and branches into four main categories: classification, regression, clustering, and dimensionality reduction. Each category contains a series of decision points (e.g., 'more data?', 'few features should be important?') that lead to specific algorithms (e.g., SVC, SGD Classifier, KNeighbors Classifier, Linear SVC, Naive Bayes, Spectral Clustering, KMeans, MiniBatch KMeans, MeanShift, VMGM, Randomized PCA, Logistic, Spectral Embedding, LLE, Kernel approximation). The flowchart also includes a 'Back' button and the 'scikit-learn' logo.</p> </div> <p>참고 https://scikit-learn.org/stable/</p>