MFCC와 딥러닝을 이용한 음성신호분류의 간단한 입문

지도교수 : 김 윤
연 구 자 : 김 성 일

< 목 차 >

요 약

2016년 ‘알파고 쇼크’ 이후로 머신러닝과 딥러닝에 대해 컴퓨터과학자와 전공 과학도들만의 관심뿐만 아니라, 일반 대중들도 큰 흥미를 끌어 인지도가 크게 높아지게 되었다. 일상생활에서도 접하는 IT기술 중에도 애플社 ‘Siri’의 음성인식기능, 현대차나 테슬라社의 자율주행기능이 바로 저 딥러닝을 이용한 기술이어서 확실한 체감 또한 존재한다. 본지에서는 그 중에서 음성신호인식과 분류에 필요한 개념들, 그리고 딥러닝의 간단한 적용사례에 대해 다루고자 한다.

주요어 : 딥러닝, MFCC

1. 서론

'음성신호' 란 공기를 매질로 하여 그 압력의 변화를 이용하는 송/수신 방법 혹은 매체이다. 하지만 실생활에서의 음성신호는 0과 1로 나타내는 이진신호가 아닌 아날로그 신호이므로, 컴퓨터과학에서는 이를 샘플링, 부호화, 양자화 등의 방법을 이용하여, 컴퓨터로 처리할 수 있는 이진 데이터로 만든 뒤 저장한다.

이와 반대로 인간 등 동물의 청각처리 능력으로는 공기 압력의 변화패턴을 감지하는 것만으로도 해당 음성신호가 어떤 것인지를 바로 분류할 수 있지만, 컴퓨터로는 특별한 과정을 단계별로 거쳐야 음성신호의 분류가 가능하다. 이를 구현하기 위해 필요한 개념들인 푸리에 변환, MFCC, 그리고 딥러닝에 대해 다음의 본론에서 서술하고자 한다.

2. 본론

2.1. 푸리에 변환

푸리에 변환(Fourier Transform)은 시간이나 공간 등의 어떠한 축선 기준에서 진폭, 주파수 등의 연속된 값이 주어졌을 때, 이를 여러 개의 사인함수나 코사인함수의 주기함수로 분해하여 표현하는 방법을 일컫는다. 프랑스의 수학자 Joseph Fourier(1768-1830)가 주창한 개념이다.

예를 들어 어떠한 노래가 공연되고 있다고 가정했을 때, 이 노래가 나타내는 음성신호만으로는 가수의 육성에 해당하는 주파수성분, 드럼연주에 해당하는 주파수성분, 기타연주에 해당하는 주파수성분이 매 구간마다 얼마나 포함되어있는지 곧바로 알 수는 없다. 이 경우 푸리에 변환 등의 해석을 거치면 각각의 주파수성분의 포함 정도를 알 수 있게 된다.

$$\int_{-\infty}^{\infty} F(k)\, e^{2\pi i k x}\, dk$$

[사진 1] 푸리에 변환 수식

푸리에 변환 수식은 위와 같다.

2.2. MFCC(Mel Frequency Cepstral Coefficient)

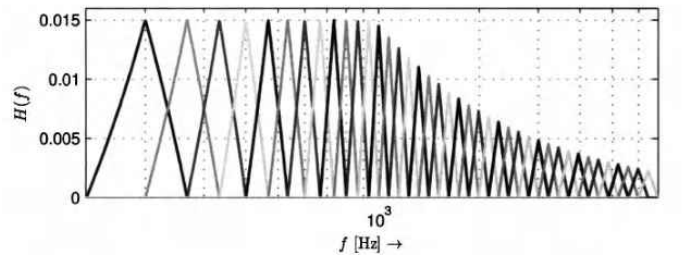
푸리에 변환은 사인함수와 코사인함수, 그리고 복소수가 쓰여서 컴퓨터를 이용한 연산에는 비효율적이다. 따라서, DCT(Descrete Cosine Transform), 즉, 코사인 함수만을 여러 개 합하여 변환하는 형태의 더욱 효율적인 기법을 사용한다. DCT는 음성신호뿐만 아니라, JPEG, MPEG 등의 사진/영상신호에도 쓰인다.

여기서 MFCC는 1980년부터 쓰이기 시작하였으며, Paul Mermelstein이 창안하였다. 음악의 음계에 해당되는 주파수들을 중심으로 삼각파형을 여러 개 만든 뒤, 원본 신호와 곱연산과 로그연산을 하고 이것을 복수의 코사인함수 합 형태로 근사 값을 만들어 각 mel(음계) 단위의 블록으로 나눠 저장하는 것이다.

다만 MFCC가 인간의 반고리관 청세포의 청각신호 수용을 모방한 것이라고 흔히 알려져 있으나, Alexander Lerch의 2012년 저서 「An Introduction to Audio Content Analysis」 에 따르면 MFCC가 신경학적 근거 위에서 만들어졌다는 증거가 없다고 밝힌바가 있다.

Table 3.8 Properties of three popular MFCC implementations

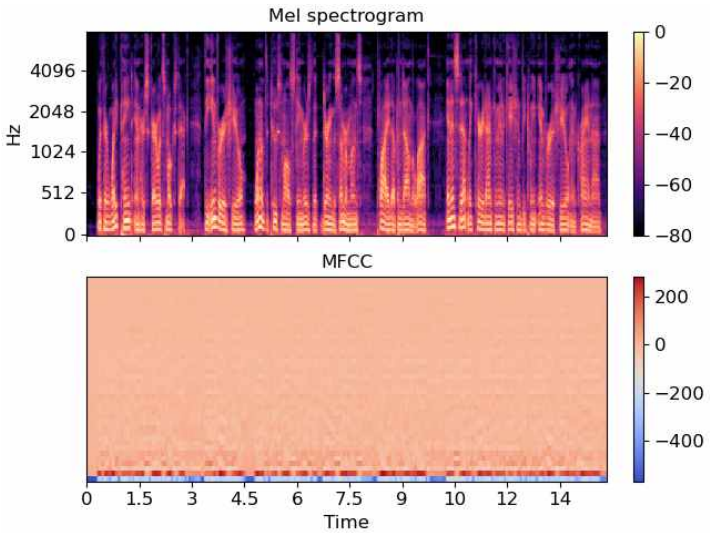
Property	*DM*	*HTK*	*SAT*
Num. filters	20	24	40
Mel scale	lin/log	log	lin/log
Freq. range	[100; 4000]	[100; 4000]	[200; 6400]
Normalization	Equal height	Equal height	Equal area



[사진 2] MFCC 삼각파 블록의 예

이 외에 STFT(국소푸리에변환) 역시 주파수성분의 분석으로 많이 쓰이나, 검출하려는 음성신호의 주파수대역을 이미 예상할 수 있는 경우에는 MFCC가 더 효율적이므로 본지에는 MFCC만을 다루었다. STFT와 MFCC에 대한 비교의 더욱 자세한 내용은 다음 링크를 참조할 수 있다.

(https://medium.com/analytics-vidhya/simplifying-audio-data-fft-stft-mfcc-for-machine-learning-and-deep-learning-443a2f962e0e)



[사진 3] MFCC 처리 후 시각화의 예

2.3. 딥러닝

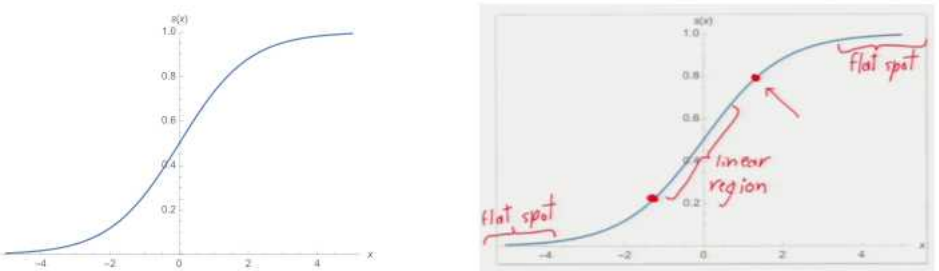
2016년 속칭 '알파고 쇼크' 이후로 일반인에게도 잘 알려진 딥러닝은, 2022년 오늘날에 더욱 강력해진 CPU와 GPU 성능과 Tensorflow, PyTorch, sklearn 등의 딥러닝용 오픈소스 소프트웨어들 덕분에 일반인들도 취미로써 즐기기에 큰 무리가 없어진 수준까지 도달해 있다.

딥러닝은 기계학습(Machine Learning)의 일부에 속하며, 특히나 생물의 신경망과 신경세포의 동작을 모방한 인공신경세포 유닛 각각과 인공신경망을 이용하는 컴퓨터의 기계학습이라고 할 수 있다. 여기서 기계학습에 대해 컴퓨터과학자 Tom Mitchell이 정의한 바에 따르면, "경험 E와 작업 T, 그리고 성능 P가 있을 때, T가 P에 의해 측정되며 E에 의해 향상되는 컴퓨터 프로그램을 일컫는다."라고 한다.

딥러닝과 머신러닝에 대한 더욱 자세한 개념은 美스탠포드 대학교 컴퓨터과학 교수 Andrew Ng가 Coursera에서 무료로 공개한 강의 링크에서 해당 내용을 참조할 수 있다. 특히, Cost Function, Gradient Descent, Back Propagation 등의 머신러닝에 관련된 다양한 기본개념들도 포함되어 유용하다.

다만, 아래의 2.4. 에서 다룰 내용은 데이터셋 각각이 어떤 분류의 것인지를 인간이 이미 알고 있으므로 'Supervised Learning' 에 속한다. 또한 위의 Andrew Ng 강의에서 Cost Function을 구하기 위해 사용된 Sigmoid Function 대신에, 2.4. 의 사용례에선 ReLU가 사용되었고 마지막엔 softmax함수가 쓰였음을 알 수 있다. softmax함수는 결과 값이 나올 수 있는 범주 각각의 확률을 나타내며, 모든 것을 더한 합은 1이 된다. 대개는 가장 높은 확률인 것을 취하여 마지막 노드 결과 값 출력을 위해서 많이 쓰인다.

Jonathan Richard Shewchuk의 집필 강의록에서 밝힌 바에 따르면, Sigmoid Function에 비해 ReLU가 다음과 같은 장점을 가진다고 한다. ReLU는 선형적 함수여서 연산속도가 빠르며 'vanishing gradient problem'을 겪을 위험이 줄어드는 점이 바로 그것이다.

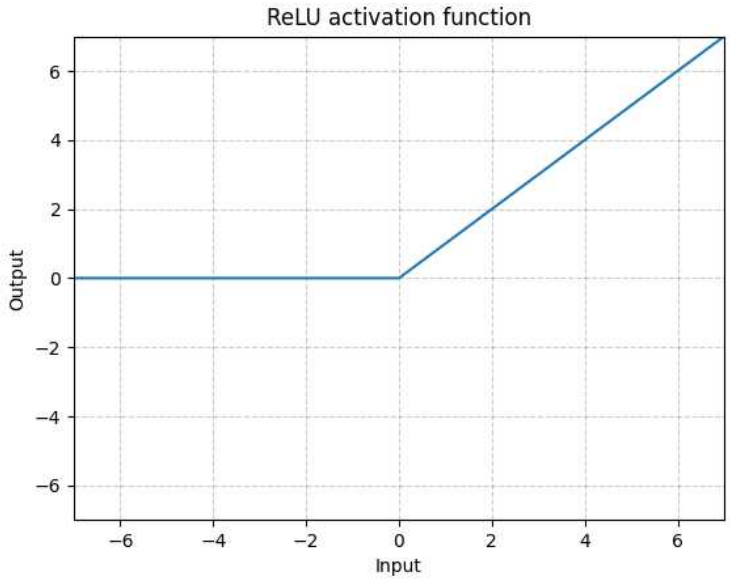


logistic.pdf [Draw flat spots, "linear" region, & maximum curvature points (at $s(\lambda) \doteq 0.21$ and $s(\lambda) \doteq 0.79$) of the sigmoid function. Ideally, we would stay away from the flat spots.]

[사진 4] Logistic Regression 함수 개형

'vanishing gradient problem' 에 대해 조금 더 살펴보자면, 위의 그림에서 알 수 있듯이 Sigmoid Function에서 쓰이는 Logistic Regression 함수에서는 x값이 일정 구간(대략 -1.5~+1.5)을 지나 y축 값이 0 혹은 1에 가까워질수록 기울기가 1 미만이 되는 경우가 생긴다.

이렇게 되면 Gradient Descent를 구하는 편미분 수식의 항 일부에서 s(1-s) 값이 0에 가까워져서 학습속도가 느려지거나 실제 최적값을 찾지 못하게 될 위험이 생긴다. 그러므로 ReLU 등의 다른 개선된 함수를 주어진 조건에 알맞게 사용하거나, 다른 수학적 기법으로 보완하는 것이 바람직하다고 한다.



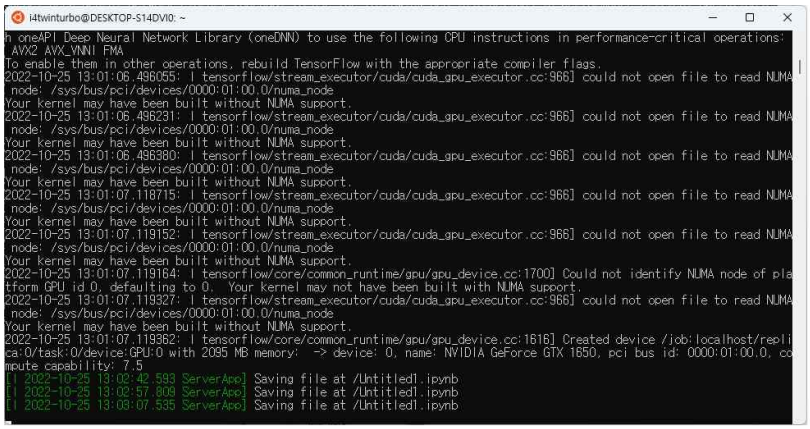
[사진 5] ReLU 함수 개형

2.4. ‘UrbanSound8K’ 데이터셋과 딥러닝을 이용한 음성신호의 분류

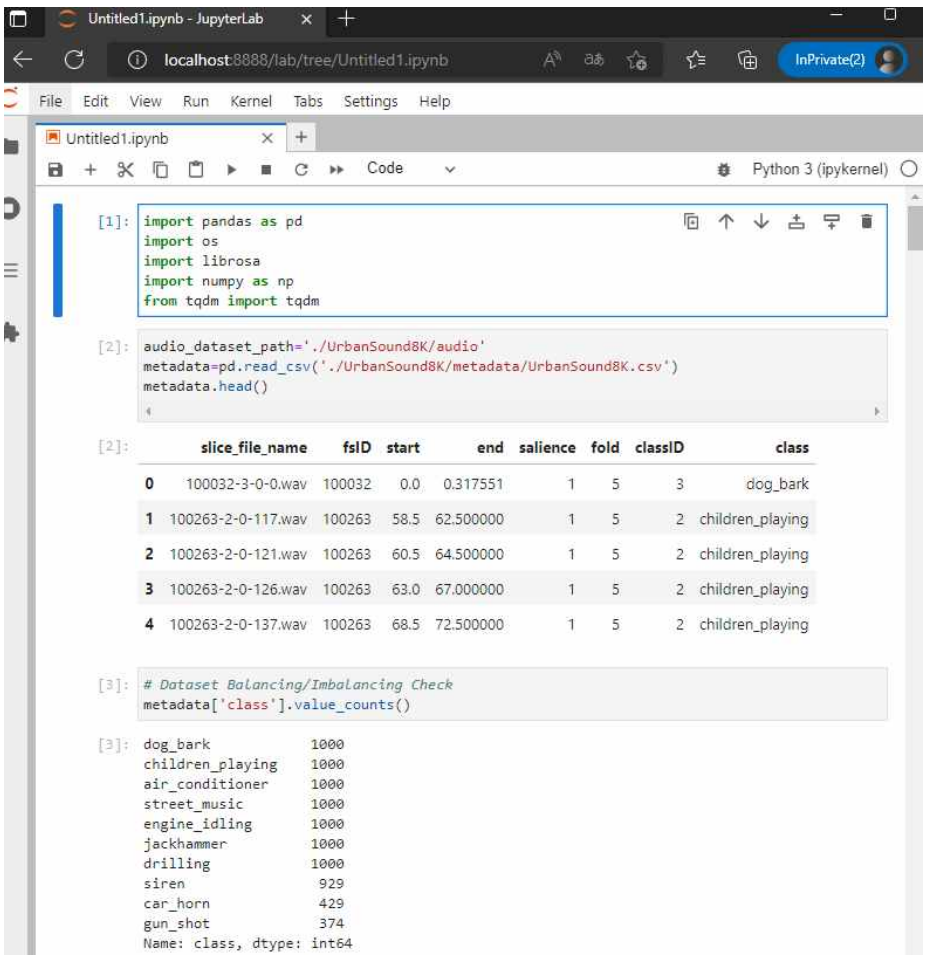
‘UrbanSound8K’ 데이터셋이란 8,732개의 wav 포맷의 음성파일들 각각에 라벨이 붙여진 공개된 데이터셋을 말한다. 여기에 첨부된 csv파일에서 각각의 wav파일들이 어떤 소리인지 분류가 되어있으며, 소리의 종류는 자동차 경적음, 총소리, 엔진소리 등의 총 10가지가 있다.

위의 ‘UrbanSound8k’는 美 뉴욕대의 Justin Salamon, Christoper Jacoby, Juan Pablo Bello가 만든 것이며, https://urbansounddataset.weebly.com/ 에서 2022년 10월 현재 무료로 제공하고 있다.

이 데이터셋을 이용해 딥러닝을 진행하는데, 伊인공지능 학자 Rosario Moscato가 github에 공개한 코드를 참고하면 빠르고 쉽게 재현할 수 있다.



[사진 6] 딥러닝 환경 구성 후, Jupyter lab 실행중의 리눅스 콘솔 캡처



[사진 7] Jupyter lab 상 화면 캡처

Jupyter lab을 실행하여 Urbansound8K 데이터셋과 Rosario Moscato가 github에 공개한 코드를 차례대로 실행하면 위와 같이 정상적으로 작동하는 화면을 볼 수 있다. github 원본은 google colab을 사용하였으나, 본인은 개인의 로컬컴퓨터에 설치하여 파일 대상경로 설정을 수정하여 코드를 실행하였다.

```
Epoch 99/100
49/55 [=========================>....] - ETA: 0s - loss: 0.2320 - accuracy: 0.92
Epoch 99: val_loss did not improve from 0.27267
55/55 [==============================] - 0s 5ms/step - loss: 0.2337 - accuracy:
val_loss: 0.2989 - val_accuracy: 0.9187
Epoch 100/100
49/55 [=========================>....] - ETA: 0s - loss: 0.2341 - accuracy: 0.91
Epoch 100: val_loss did not improve from 0.27267
55/55 [==============================] - 0s 5ms/step - loss: 0.2395 - accuracy:
val_loss: 0.2862 - val_accuracy: 0.9181
Training completed in time:  0:00:30.953321
```

```
[28]: test_accuracy=model.evaluate(X_test,y_test,verbose=0)
      print(test_accuracy[1])
```

```
0.9181454181671143
```

```
[29]: filename="./196076-2-0-0.wav"
      audio, sample_rate = librosa.load(filename, res_type='kaiser_fast')
      mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
      mfccs_scaled_features = np.mean(mfccs_features.T,axis=0)
```

```
[30]: mfccs_scaled_features.shape
```

```
[30]: (40,)
```

```
[31]: mfccs_scaled_features=mfccs_scaled_features.reshape(1,-1)
      print(mfccs_scaled_features.shape)
      predicted_label = np.argmax(model.predict(mfccs_scaled_features), axis=-1)
      print('Predicted Label:',predicted_label)
      prediction_class = labelencoder.inverse_transform(predicted_label)
      prediction_class[0]
```

```
(1, 40)
1/1 [==============================] - 0s 44ms/step
Predicted Label: [2]
[31]: 'children_playing'
```

```
[ ]:
```

[사진 8] Jupyter lab 상 화면 캡처 두번째

MFCC화가 완료되고 학습이 정상적으로 작동된 경우, [29]번 입력문에서 'filename=' 이 적힌 첫 행에서 임의의 파일을 지정하여 [31]번 입력문을 실행하면 딥러닝으로 학습된 인공신경망에 의해 음성파일이 판독된 결과가 [31]번 출력문에서 정상출력되는 것을 확인할 수 있었다.

3. 결론

음성신호를 분류하기 위해 필요한 푸리에변환, MFCC 등의 기본 개념들과 딥러닝의 정의, 그리고 예시 코드에 대해 위의 본문에서 살펴보았다.

이를 이용하여 음성신호 파일이 주어졌을 때, 2022년 현재의 일반인들이 쉽게 구할 수 있는 하드웨어와 소프트웨어 자원으로도 얼마든지 딥러닝을 수행하여 이를 분류하는 프로그램을 구현할 수 있음을 알 수 있었다.

본지의 주제가 되는 음성신호 분류 기술을 응용하여 산업현장에서는 비정상적인 기계소음(고장음)을 탐지하고, 가정에서는 아기 울음소리, 가스 누출음과 같은 경계해야 할 소음을 탐지하는 등의 기능이 탑재된 음성인식 탐지기/경보기와 같은 활용이 미래에 기대된다.

참고문헌

[1] Alexander Lerch. 「An Introduction to Audio Content Analysis」 pages 7-8, 51-53. IEEE, 2012.
[2] David Gerhard. 「Audio Signal Classification: History and Current Techniques」 Technical Report TR-CS 2003-07, 2003.
[3] Zvi Kons, Orith Toledo-Ronen. 「Audio Event Classification Using Deep Neural Networks」 IBM Research, 2013.
[4] https://librosa.org/doc/0.9.1/generated/librosa.feature.mfcc.html
[5] https://www.coursera.org/specializations/machine-learning-introduction
[6] Jonathan Richard Shewchuk. 「Concise Machine Learning」 pages 104-107. University of California at Berkeley, 2022.