卒 業 論 文

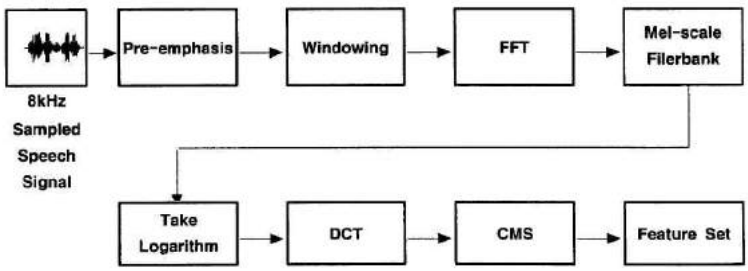
MFCC(Mel-frequency Cepstrum Coefficients) 특성 추출 기반의 화자가 부르기 적합한 곡 추천 시스템

목 차

1. 서론

본 논문의 목표는 기계학습을 이용해 사용자의 음성 특성에 맞는 적절한 곡을 추천해주는 시스템을 구현하는 것이다. 기존의 곡 추천 기법들은 상황 인식[1], 사용자 행위[2], 곡의 장르나 가사의 감성[3][4] 분석을 이용한 추천 시스템으로 주로 사용자의 감정(Emotion)과 곡의 분위기(Mood)에 기반을 두었다. 그리고 협업 필터링(Collaboratie filtering)과 컨텐츠 필터링(content filtering) 기법을 이용해 사용자가 듣고 싶어 하는 곡을 추천해주는 시스템이 주를 이루었다.[5][6] 이처럼 사용자가 들으려는 곡의 추천 시스템 기술은 빠르고 다양하게 발전하고 있지만, 아직까지 부르기 적합한 곡 추천 시스템 개발은 미흡한 실정이다. 이에 본 논문에서는 화자 음성발화의 주파수 특성을 수치적으로 분석한 MFCC와 기계학습을 이용해, 개개인의 음색에 따라 부르기 적합한 곡 추천 시스템을 구현하고자 한다.

음성 특성을 추출하는 방법으로는 일반적으로 LPC(Linear Predictive Coding)와 MFCC(Mel-frequency cepstrum coefficients) 계수를 사용한다. LPC 계수 추출법은 음성의 발화를 사람의 성도를 모델링하는 전달함수에 의해 발생되는 과정으로 가정하는 방법이다. 이에 반해 MFCC는 입력되어진 음성의 형태를 사람의 청각기관으로 모델링해서 변환시키는 음성에 대한 특징추출의 한 형태를 말한다. 지금의 연구결과로 MFCC에 사용되는 Cepstrum 계수를 사용하는 것이 LPC 계수에 비해 인식성능이 좋다고 발표되고 있고, 특히 주변 잡음과 채널 왜곡에 강하다고 알려져 있다.



MFCC 전체 Block Diagram

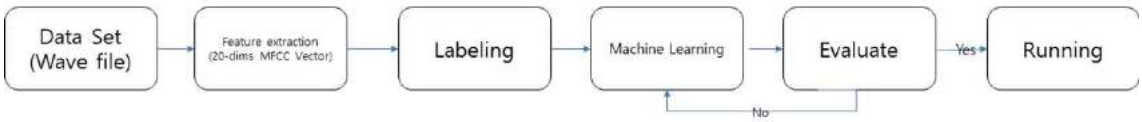
본 논문에서는 화자의 음성발화로부터 추출한 MFCC 계수와 선택한 곡 데이터를 기반으로, 뉴럴 네트워크를 이용한 곡 추천 알고리즘을 제안하고자 한다. 그리고 이 알고리즘의 한계와 향후 발전 방향을 모색해 볼 것이다.

2. 본론

2.1 연구의 내용

음성발화로부터 추출한 MFCC 계수와 선택한 곡 Data set을 만들기 위하여 Python의 librosa 라이브러리와, 뉴럴 네트워크를 이용한 곡 추천 알고리즘을 구성하기 위하여 Tensorflow를 이용하였다. [7]

본 논문이 제안하는 추천 시스템 알고리즘은 Wave file로 Data Set을 수집한 후에 20차 MFCC Vector를 추출, 각 음성에 맞는 Data를 26개의 가수 카테고리로 분류하여 Labeling한 후, Tensorflow를 이용한 Machine Learning을 통해 곡을 추천하는 구조를 갖고 있다. 알고리즘의 개략도는 다음 그림과 같다. [8]



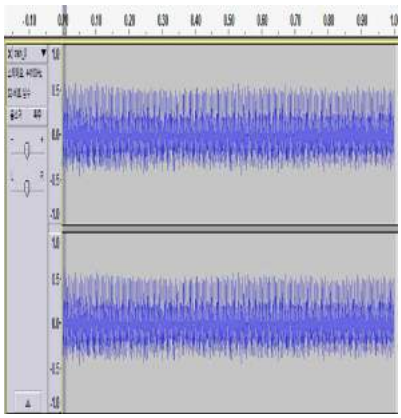
Machine Learning을 이용한 곡 추천 알고리즘

Wave file로 수집한 Data Set의 일부와 26개의 가수 카테고리, Date Set의 파형, MFCC 계수를 추출하는 코드 구성은 다음과 같다. 가수 목록은 노래방 음원차트 Top100 중에서 본 연구에 적합한 장르의 가수로 선정하였다.

Data Set

1. 엠씨더맥스	15. 거미
2. 한동근	16. 홍진영
3. 임창정	17. 불빨간사춘기
4. 정준일	18. 아이유
5. 버즈	19. 윤하
6. 김범수	20. 다비치
7. 허각	21. 태연
8. 박효신	22. 벤
9. 이적	23. 이선희
10. 김동률	24. 자우림
11. 윤도현	25. 백아연
12. 로이킴	26. 박정현
13. 버스커버스커	
14. 성시경	

26개의 가수 카테고리



Date Set의 파형

```
import librosa
import numpy as np

def extract_feature(file_name):
 X, sample_rate = librosa.load(file_name)
 mfccs = librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=20,
                              hop_length=int(sample_rate*0.01), n_fft=int(sample_rate*0.02)).T
 return mfccs
wav_files = []
all_mfccs = np.ndarray(shape=[0, 20], dtype=np.float32)
```

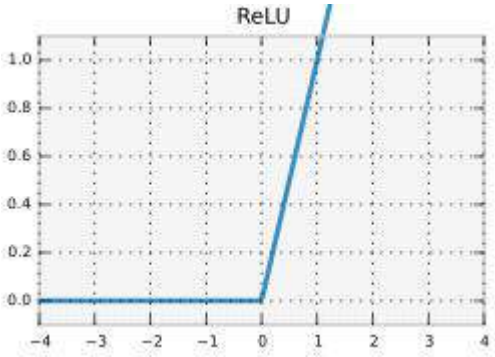
MFCC 계수 추출

음성 Data set을 각 1초 구간으로 편집 한 후 위와 같은 librosa 라이브러리에서 mfcc 추출 함수를 이용해 20차 벡터를 추출한 결과 프레임별로 각 100개정도의 벡터가 추출되었고, numpy 라이브러리를 이용해 평균값을 구해 최종적으로 222*20 array를 all_mfccs에 저장하였다. 결과는 아래와 같다.

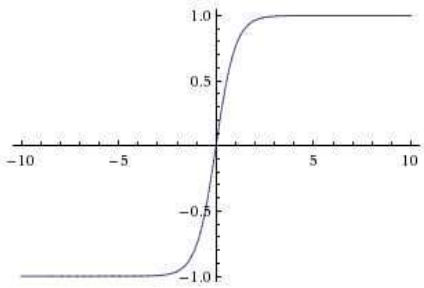
```
[[-271.83684776  170.68912159  -38.4202101  ...,  -14.96087148
    5.06767482    3.59391386]
 [-279.69699427  180.07311726  -45.38507182 ...,   -9.03179248
    2.83350918    0.97250256]
 [-283.09736221  177.38934725  -47.8387029  ...,   -5.99317605
    1.72218903   -3.8391016 ]
 ...,
 [-283.67391669  160.23879701  -33.96792819 ...,   -2.62044713
  -14.76747587   -6.82265772]
 [-278.42872023  155.47188314  -32.76265418 ...,   -4.00883777
  -16.69055942   -4.39345042]]
```

20차 MFCC 벡터

이렇게 추출된 20차 MFCC 벡터를 Input, 그에 맞는 26개의 가수 카테고리를 Output으로 Tensorflow를 이용하여 학습시켰다. 본 연구에서 Machine Learning에 사용한 히든 레이어는 2개로, 1차는 sigmoid보다 성능이 좋은 ReLu함수를, 2차는 히든레이어에서 좀 더 많은 변화폭을 보장하는 하이퍼볼릭 탄젠트함수를 활성함수로 사용하였다. 또한 2차 레이어에서는 50%의 weight값들을 무시하는 Dropout을 사용하였는데 overfitting에 대한 해결책으로 사용할 수 있는 방법이다.



ReLu 함수



하이퍼볼릭 탄젠트 함수

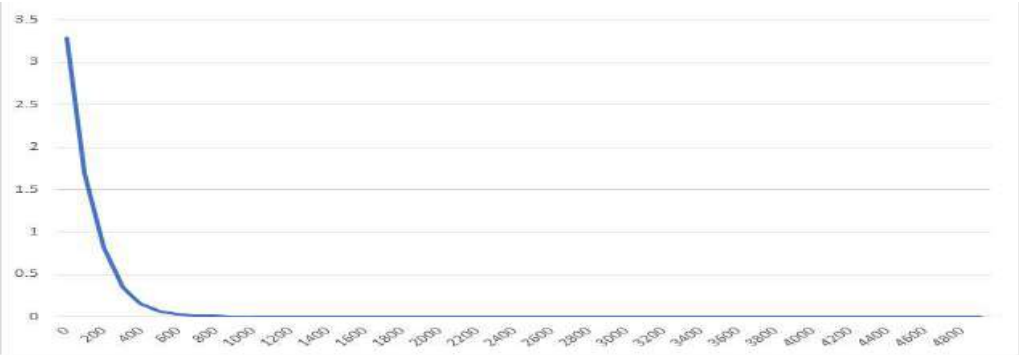
Tensorflow의 학습기법에 대해 간략한 설명을 덧붙이면 우선 Cost Function은 Labeling된 학습데이터에 대하여 학습과정에서 생성한 결과값과 기댓값의 차이를 계산하는 함수이다. Tensor Flow는 Optimizer Algorithm을 통하여 Cost Function의 값이 최소가 되도록 Back propagation을 수행한다. Cost Function으로는 Cross Entropy Function을 사용하였다. Cross Entropy Function은 MSE함수보다 빠른 학습속도를 보장하기에 이를 택하였다. 또한 Optimize과정에서는 Adam Optimizer를 사용하였는데, 많이 사용되는 방법인 경사 하강법에 비하여 높은 정확도를 보장해 준다. 각 Layer의 코드 구성은 다음과 같다.

```
oo = tf.truncated_normal([n_dim, n_hid1], mean=0.0, stddev=sd)
W_1 = tf.Variable(oo)
b_1 = tf.Variable(tf.constant(value=0.1, shape=[n_hid1]), name="b1")
h_1 = tf.nn.relu(tf.matmul(X, W_1) + b_1)

W_2 = tf.Variable(tf.truncated_normal([n_hid1, n_hid2], mean=0, stddev=sd), name="w2")
b_2 = tf.Variable(tf.constant(value=0.1, shape=[n_hid2]), name="b2")
h_2 = tf.nn.tanh(tf.matmul(h_1, W_2) + b_2)
h_2_drop = tf.nn.dropout(h_2, keep_prob)

W = tf.Variable(tf.truncated_normal([n_hid2, n_classes], mean=0, stddev=sd), name="w")
b = tf.Variable(tf.constant(value=0.1, shape=[n_classes]), name="b")
y_hypothesis = tf.nn.softmax(tf.matmul(h_2_drop, W) + b)
prediction = tf.argmax(y_hypothesis,1)
y_ = tf.matmul(h_2_drop, W) + b
sess = tf.InteractiveSession()


cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels= Y, logits= y_))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
saver = tf.train.Saver()
sess.run(tf.global_variables_initializer())
```

각 Layer의 코드 구성

2.2 실험 결과

학습 횟수는 총 5000회이며 100회마다 체크되도록 설정을 하였다. Loss의 값은 처음에 3.639963이였지만 점차 줄어서 마지막에는 0.109523이라는 값이 나오게 되었다.



5000번 학습 시의 Cross Entropy Loss의 감소

다음은 Tensorflow를 이용한 학습 및 테스트에서 처음과 마지막 단계이다.

```
step 0, loss 3.639963, training accuracy 0.0135135
prediction :  [ 9  9  9  9  9  9  9  9  9  9 19  9 19]
label:  [15 16 17  0  5  8  0  3  4  5 10 11 19  6]
step 0, test accuracy 0.000000
The model is saved in file as, ./mymodel1.ckpt:
step 100, loss 2.327356, training accuracy 0.378378
prediction :  [17 17 16 12 10 12 18 18 14  5 18 18 20 10]
label:  [15 16 17  0  5  8  0  3  4  5 10 11 19  6]
step 100, test accuracy 0.071429
The model is saved in file as, ./mymodel1.ckpt:
step 200, loss 1.894224, training accuracy 0.527027
prediction :  [17 17 18 12 10 10  0 18  4  5 18 18 20  5]
label:  [15 16 17  0  5  8  0  3  4  5 10 11 19  6]
step 200, test accuracy 0.214286
The model is saved in file as, ./mymodel1.ckpt:
step 300, loss 1.545411, training accuracy 0.675676
prediction :  [17 18 18 12 10  3  0  4  4  5 18 18 20  6]
label:  [15 16 17  0  5  8  0  3  4  5 10 11 19  6]
step 300, test accuracy 0.285714
The model is saved in file as, ./mymodel1.ckpt:
```

학습의 첫 단계

```
label:  [15 16 17  0  5  8  0  3  4  5 10 11 19  6]
step 4600, test accuracy 0.357143
The model is saved in file as, ./mymodel1.ckpt:
step 4700, loss 0.115447, training accuracy 0.995496
prediction :  [18 18 18 14 10 10  0  3  4  5 19 15 18  6]
label:  [15 16 17  0  5  8  0  3  4  5 10 11 19  6]
step 4700, test accuracy 0.357143
The model is saved in file as, ./mymodel1.ckpt:
step 4800, loss 0.113207, training accuracy 0.995496
prediction :  [18 18 18 14 10 11  0  3  4  5 19 15 18  6]
label:  [15 16 17  0  5  8  0  3  4  5 10 11 19  6]
step 4800, test accuracy 0.357143
The model is saved in file as, ./mymodel1.ckpt:
step 4900, loss 0.109523, training accuracy 0.995496
prediction :  [18 18 18 14 10 11  0  3  4  5 19 15 18  6]
label:  [15 16 17  0  5  8  0  3  4  5 10 11 19  6]
step 4900, test accuracy 0.357143
The model is saved in file as, ./mymodel1.ckpt:
2017-06-15 21:28:30.458875: I c:₩tf_jenkins₩home₩workspac
Answer is  [10]
```

학습의 마지막 단계

학습의 정확도를 측정을 해 보기 위하여 Data Set으로 수집을 하였던 같은 목소리의 다른 음성으로 Test를 다시 진행을 해 보았다. Test data로 Test를 한 결과 Test accuracy는 5000번 학습 시 0.357143으로 14개의 Test중 정답인 label과 예측값인 prediction을 비교하였을 때, 대략 5개정도 맞춘 값으로 수렴한다. 이 결과를 보면 어떠한 목소리로 학습을 하였을 때, 그 목소리의 다른 상태의 음성으로 Test를 하면 정확도가 0.357143정도 나온다는 것을 의미한다. 또한, 새로운 어떤 임의의 음성을 이 모듈에 넣었을 때 10이라는 결과가 나오고 이 값을 통해서 11번째 '윤도현'이라는 가수를 추천해주는 결과를 확인 할 수 있다.

3. 결론 및 향방

본 논문에서는 Python의 Library와 Tensorflow를 이용하여 학습하는 모듈을 이용해 곡 추천 알고리즘을 구현하였다. 앞서 서론에서 밝혔듯 본 논문에서는 MFCC 계수와 선택한 곡 데이터를 기반으로 곡 추천 알고리즘을 구현하였고, 총 222개의 목소리의 데이터를 바탕으로 곡을 추천을 하는 결과를 얻을 수 있었다.

위 결과를 통해 사람들의 목소리를 바탕으로 어떤 특정 목소리를 입력하였을 때 그 목소리와 가장 잘 맞는 곡을 추천하는 것을 확인 할 수 있다. 하지만 본 논문에서 사용한 알고리즘은 총 222 개의 목소리라는 적은 양의 데이터밖에 없어서 학습 모듈의 높은 정확도를 얻기는 부족하였다. 또한 음성을 추출하는 과정에서 잡음과 녹음환경, 발성법, 음정에 따라서 소리가 다르기 때문에 아직까지 보완해야 할 다양한 변수가 존재했다.

음성인식 기술은 다양한 방법으로 연구되고 있으며 특히 사용자 인증, 성별 및 연령 구별, 감정 분석 등 여러 분야에 활용되고 있다.[9] 이 연구 결과를 바탕으로 다양하고 많은 데이터의 수집과, 사람의 고유한 목소리가 들어간 깔끔한 음성을 얻고 화성과 발성, 장르 등 음성학적 내용을 보태면 보다 더 정확한 곡 추천 알고리즘을 구현할 수 있을 것으로 예상된다.

참 고 문 헌

[1] 상황 인식을 이용한 사례기반 음악추천시스템. 이재식 이진천. 2006.

[2] 사용자 행위에 기반을 둔 개인 맞춤형 음악 추천 시스템의 설계. 배재희 김현경 김영인. 2015

[3] 사용자 감성과 음원 무드 기반 음악 추천 시스템. 최현석 외 6명. 2010.

[4] 감성 기반 음악 검색 및 추천 시스템 설계. 윤보국 홍성용. 2011.

[5] Hybrid Music Recommendation System. Intekhab Naser. 2014

[6] Collaborative Deep Learning for Recommender Systems. Hao Wang. 2015

[7] 적응 MFCC와 Neural Network 기반의 음성인식법. 배현수 이석규. 2010.

[8] Tensor Flow를 이용한 음향인식기 제작. 오준석 김기환. 2016.

[9] 음성 확인기술을 이용한 전자상거래용 사용자 인증 시스템 개발. 서경대학교. 2000.